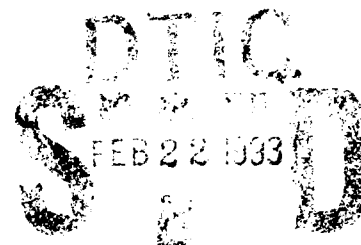




2

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

ARMA MODELING OF SIGNALS
IN
THE TIME DOMAIN

by

Carlos H. Velasco

December, 1992

Thesis Advisor:
Second reader:

Charles W. Therrien
Murali Tummala

Approved for public release; distribution is unlimited.

93-03610



REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)				
6a. NAME OF PERFORMING ORGANIZ. Naval Postgraduate School	6b. OFFICE SYMBOL (if applicable) EC	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
				WORK UNIT ACCESSION N
11. TITLE (Include Security Classification) ARMA MODELING OF SIGNALS IN THE TIME DOMAIN				
12. PERSONAL AUTHOR(S) Velasco, Carlos H.				
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (YY, MM, DD) December 1992	15. PAGE COUNT 76	
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	Iterative Prony. ARMA Modeling. Acoustic Modeling. Iterative Pre-filtering. Time Domain Modeling.	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This thesis develops an iterative algorithm for the design of ARMA models of signals in the time domain. The algorithm is based on optimization techniques, particularly a gradient technique known as the <i>restricted step method</i> is used. The new algorithm is called the <i>iterative Prony method</i> , and the results obtained using this new method are compared to those obtained using the iterative prefiltering algorithm. The thesis shows that the performance of the iterative Prony method is in most of the cases comparable or superior to that of the iterative prefiltering algorithm.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASS./UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Therrien, Charles W.			22b. TELEPHONE (Incl. Area Code) (408) 546-3347	22c. OFFICE SYMBOL EC/Ti

Approved for public release; distribution is unlimited.

**ARMA MODELING OF SIGNALS
IN THE TIME DOMAIN**

by

Carlos Hernando Velasco Solano
Lieutenant Commander, Colombian Navy
B.S.E.E., Colombian Naval Academy, 1986

Submitted in partial fulfillment of the
requirements for the degree of

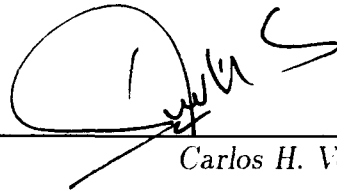
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

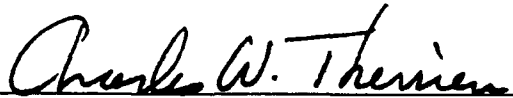
December, 1992

Author:

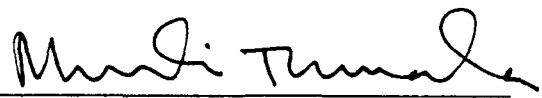


Carlos H. Velasco S.

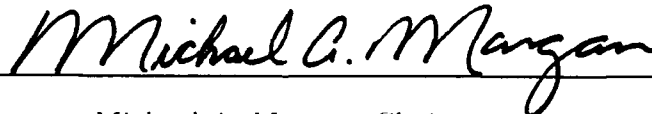
Approved By:



Charles W. Therrien, Thesis Advisor



Murali Tummala, Second Reader



Michael A. Morgan, Chairman.

Department of Electrical & Computer Engineering

ABSTRACT

This thesis develops an iterative algorithm for the design of ARMA models of signals in the time domain. The algorithm is based on optimization techniques, particularly a gradient technique known as the *restricted step method* is used. The new algorithm is called the *iterative Prony method*, and the results obtained using this new method are compared to those obtained using the iterative prefiltering algorithm. The thesis shows that the performance of the iterative Prony method is in most of the cases comparable or superior to that of the iterative prefiltering algorithm.

DISQUALIFY FROM THE

Accession For	
NTIS ORIGIN	<input checked="" type="checkbox"/>
DATE TAG	<input type="checkbox"/>
USE REQUIRED	<input type="checkbox"/>
JUSTIFICATION	

TABLE OF CONTENTS

I. INTRODUCTION	1
A. THE IDEA OF ARMA MODELING	1
B. WHY AN ITERATIVE ALGORITHM IN THE SIGNAL DOMAIN	2
C. THESIS OUTLINE	3
II. ARMA MODELING OF SIGNALS	4
A. MODELING METHODS USED IN THIS THESIS	4
B. OVERVIEW OF MODELING METHODS	5
1. Prony's Method	5
2. Signal Domain Form of Prony's Method	8
3. Iterative Prefiltering	10
III. ITERATIVE ALGORITHM IN THE SIGNAL DOMAIN	14
A. MULTIDIMENSIONAL OPTIMIZATION BY GRADIENT METHODS	14
B. THE ITERATIVE PRONY METHOD	17
1. Vector \mathbf{g} of first derivatives	19
2. Matrix \mathbf{G} of second derivatives	22
3. Algorithm implementation	24
IV. PERFORMANCE OF THE ALGORITHM AND MODELING RESULTS	27
A. TEST DATA USED IN THIS THESIS	27
B. PRESENTATION OF RESULTS	27
1. Simulated test data	29
2. Acoustic test data	40
V. CONCLUSIONS	58

A. DISCUSSION OF RESULTS	58
B. RECOMMENDATIONS FOR FUTURE WORK	59
LIST OF REFERENCES	60
INITIAL DISTRIBUTION LIST	62

LIST OF TABLES

4.1	DESCRIPTION OF <i>SIMULATED</i> TEST DATA	28
4.2	DESCRIPTION OF <i>ACOUSTIC</i> TEST DATA	28
4.3	POLE-ZERO LOCATION OF THE SYSTEMS USED TO MODEL THE <i>SIMULATED</i> NOISY TEST DATA	41

LIST OF FIGURES

2.1	Block diagram for the direct method for signal modeling.	5
2.2	Block diagram for the indirect modeling problem.	8
2.3	Block diagram for the iterative prefiltering method.	11
2.4	Signal <i>wren01</i> and its 4 poles/4 zeros models. (a) Using Prony's method. (b) Using Signal Domain form of Prony's method. (c) Using Iterative Prefiltering.	13
3.1	Displacement of the poles and zeros of an iterative Prony's model . .	25
3.2	Displacement of the poles and zeros of a 4-3 model. The modeled signal in this case actually has two poles on the real axis	26
3.3	Displacement of the poles and zeros of a 4-3 model. The initial model shows two poles in the real axis, the final model in this case has no poles in the axis	26
4.1	Signal <i>t01_n</i> and its 2 poles-1 zero models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>t01_n</i> and the iterative prefiltering model. (c) Signal <i>t01_n</i> and the iterative Prony model.	30
4.2	Signal <i>t01_n</i> and its 4 poles-3 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>t01_n</i> and the iterative prefiltering model. (c) Signal <i>t01_n</i> and the iterative Prony model.	31

4.3	Signal $t02_n$ and its 4 poles-3 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t02_n$ and the iterative prefiltering model. (c) Signal $t02_n$ and the iterative Prony model.	32
4.4	Signal $t02_n$ and its 6 poles-5 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t02_n$ and the iterative prefiltering model. (c) Signal $t02_n$ and the iterative Prony model.	33
4.5	Signal $t03_n$ and its 4 poles-3 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t03_n$ and the iterative prefiltering model. (c) Signal $t03_n$ and the iterative Prony model.	34
4.6	Signal $t03_n$ and its 6 poles-5 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t03_n$ and the iterative prefiltering model. (c) Signal $t03_n$ and the iterative Prony model.	35
4.7	Signal $t04_n$ and its 4 poles-3 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t04_n$ and the iterative prefiltering model. (c) Signal $t04_n$ and the iterative Prony model.	36
4.8	Signal $t04_n$ and its 6 poles-5 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t04_n$ and the iterative prefiltering model. (c) Signal $t04_n$ and the iterative Prony model.	37

4.9	Signal <i>t05_n</i> and its 2 poles-1 zero models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>t05_n</i> and the iterative prefiltering model. (c) Signal <i>t05_n</i> and the iterative Prony model.	38
4.10	Signal <i>t05_n</i> and its 4 poles-3 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>t05_n</i> and the iterative prefiltering model. (c) Signal <i>t05_n</i> and the iterative Prony model.	39
4.11	Signal <i>rowcLa</i> being modeled by iterative prefiltering using a (10,9) order system. (a) Normalized squared-norm of the error between the model and the actual signal. (b) Signal <i>rowcLa</i> and the iterative prefiltering model selected from the 20 th iteration. (c) Signal <i>rowcLa</i> and the iterative prefiltering model selected from the 17 th iteration. . .	43
4.12	Behavior of the poles of the system used to model the signal <i>rowcLa</i> during one oscillation of the iterative prefiltering algorithm. (a) 15 th iteration. (b) 16 th iteration. (c) 17 th iteration. (d) 18 th iteration. (e) 19 th iteration. (f) 20 th iteration.	44
4.13	Behavior of the zeros of the system used to model the signal <i>rowcLa</i> during one oscillation of the iterative prefiltering algorithm. (a) 15 th iteration. (b) 16 th iteration. (c) 17 th iteration. (d) 18 th iteration. (e) 19 th iteration. (f) 20 th iteration.	45
4.14	Signal <i>wren01</i> and its 7 poles-6 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>wren01</i> and the iterative prefiltering model. (c) Signal <i>wren01</i> and the iterative Prony model.	48

4.15	Signal <i>wren01</i> and its 12 poles-11 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>wren01</i> and the iterative prefiltering model. (c) Signal <i>wren01</i> and the iterative Prony model.	49
4.16	Signal <i>vowel_a</i> and its 10 poles-9 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>vowel_a</i> and the iterative prefiltering model. (c) Signal <i>vowel_a</i> and the iterative Prony model.	50
4.17	Signal <i>vowel_a</i> and its 14 poles-13 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>vowel_a</i> and the iterative prefiltering model. (c) Signal <i>vowel_a</i> and the iterative Prony model.	51
4.18	Signal <i>bio_2133a</i> and its 8 poles-7 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>bio_2133a</i> and the iterative prefiltering model. (c) Signal <i>bio_2133a</i> and the iterative Prony model.	52
4.19	Signal <i>bio_2133a</i> and its 12 poles-11 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>bio_2133a</i> and the iterative prefiltering model. (c) Signal <i>bio_2133a</i> and the iterative Prony model.	53
4.20	Signal <i>bio_2385a</i> and its 8 poles-7 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>bio_2385a</i> and the iterative prefiltering model. (c) Signal <i>bio_2385a</i> and the iterative Prony model.	54

4.21	Signal <i>bio_2385a</i> and its 12 poles-11 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>bio_2385a</i> and the iterative prefiltering model. (c) Signal <i>bio_2385a</i> and the iterative Prony model.	55
4.22	Signal <i>bio_80a</i> and its 8 poles-7 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>bio_80a</i> and the iterative prefiltering model. (c) Signal <i>bio_80a</i> and the iterative Prony model.	56
4.23	Signal <i>bio_80a</i> and its 12 poles-11 zeros models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal <i>bio_80a</i> and the iterative prefiltering model. (c) Signal <i>bio_80a</i> and the iterative Prony model.	57

ACKNOWLEDGMENT

To my children, Carlos Andres, Diego Felipe , and Maria Carolina whose constant efforts to prevent me from finishing this thesis fortunately failed. To my wife, Monica who deserves most of the credit for this work. Only her love, dedication, inspiration, and support during these last two years made all this possible. I am also indebted to my thesis advisor, Dr. Charles W. Therrien, for introducing me to the fields of signal processing and for pointing me in the right direction in the completion of this work.

I. INTRODUCTION

A. THE IDEA OF ARMA MODELING

The goal of *linear modeling* is to accurately represent an observed data sequence as the output of a linear filter. The idea of representing a complicated process with a comparatively simpler model has many different applications. Curve fitting in mathematical modeling, analysis of electronic devices using equivalent circuits, and system transfer functions in automatic control are just a few examples outside the area of digital signal processing. Parametric modeling has also a large number of applications in signal processing. Currently there is considerable interest in the parametric modeling approach to spectral estimation. In speech processing the applications include digital transmission, storage, and synthesis of the speech signal. Our particular interest is the modeling of sonar signals, such as biologics and other underwater acoustic data. This work forms part of an overall research program in sonar signal modeling. The research will help to understand the relative benefits of signal domain algorithms versus algorithms based on coefficients of the transfer function. It is hoped that the method developed in this thesis will become an important tool in the overall effort for sonar signal modeling.

In linear modeling the filter used to generate the data sequence is usually represented by a linear difference equation with constant coefficients. The z -transform of this type of system is a rational polynomial function. Three type of models are derived from this kind of systems: they are known as autoregressive (AR), moving average (MA), and autoregressive moving average (ARMA).

Much work has been done on AR models, which correspond to all-pole systems. The reason for that is that the parameters for the model can be obtained by solving

linear equations, and a great body of theory has been developed that applies to this problem [Ref. 1, 2]. Relatively much less work has been done with MA and ARMA models. However, since MA models have limited applications and are almost as difficult to obtain as ARMA models, most interest centers on the latter. The filter in this type of models has both poles and zeros, and the design fundamentally involves nonlinear equations. A properly designed ARMA model can provide better performance than an AR model, with a smaller number of parameters. ARMA modeling is the topic of this thesis.

B. WHY AN ITERATIVE ALGORITHM IN THE SIGNAL DOMAIN

There are many different approaches to the problem of ARMA modeling. The majority of them are based on statistical techniques [Ref. 3]. Some of these methods regard the data as a realization of a random process while others focus on the data as given [Ref. 1]. Data oriented methods try to minimize some criterion that estimates how well the model fits the data, in most cases the least squares error between the signal and the model. Stochastic approaches may attempt to estimate the model parameters directly from the data by solving nonlinear equations or by spectral factorization. The maximum likelihood procedure, for example [Ref. 1, 4], is essentially nonlinear. A number of indirect methods have been developed that modify the norm of the error by separating the AR and MA parts of the problem so that at least some of the equations to estimate the parameters become linear. This approach is found in procedures such as the Prony's method, Shank's method, and the least squares modified Yule-Walker method [Ref. 1, 2, 5, 6]. A different type of approach replaces the nonlinear problem with iteration while trying to solve for the AR and MA parameters simultaneously. The iterative prefiltering method of Steiglitz and McBride [Ref. 7, 8] is of this type.

Our method is also of the latter type. However, the advantage is that it works directly with the poles of the rational model, which we know affect the performance of the system. The poles are displaced in specific directions so that the new model minimizes the error between the model output and the original signal. Iterative prefiltering on the other hand works with the coefficients of the transfer function, so it is difficult or impossible to predict its effects on the poles and zeros of the system. The new algorithm is much more dependable with respect to convergence than the iterative prefiltering algorithm because it moves poles and zeros specifically to minimize the error between the model and the original signal.

C. THESIS OUTLINE

The remainder of this thesis is organized as follows. Chapter II presents the modeling methods that are used in this thesis and gives a brief explanation of all of them. Chapter III introduces the reader to the theory of multidimensional optimization by gradient methods and develops the iterative Prony method, which is the main contribution of this thesis. Chapter IV presents the results of testing the algorithm on simulated and real acoustic data and compares these results with those obtained using iterative prefiltering. Chapter V gives conclusions and suggestions for future research.

II. ARMA MODELING OF SIGNALS

A. MODELING METHODS USED IN THIS THESIS

This thesis deals with deterministic approaches to ARMA modeling. Two types of modeling methods are considered. First we have non-iterative methods like Prony's method and its alternate signal domain form [Ref. 1]; second we consider iterative methods like iterative prefiltering and the new *iterative Prony* method developed in this thesis.

The goal of Prony's method is to represent a given sequence $x[n]$ as the impulse response of a linear time invariant (LTI) system. In the transform (z) domain this representation has the form

$$X(z) \approx \frac{B(z)}{A(z)} \quad (2.1)$$

where $X(z)$ is the z -transform of $x[n]$ and $B(z)/A(z)$ represents the transfer function of the system. This approximation is explained in more detail in the next section. What has become known as Prony's method in the current signal processing literature differs from Prony's original work in some respects. Our basic form of Prony's method solves for the coefficients of the transfer function in (2.1).

The signal domain form of Prony's method is closer to Prony's original work [Ref. 1, p.560] and seeks to represent the data in terms of a set of damped exponentials as

$$x[n] \approx c_1 r_1^n + c_2 r_2^n + \cdots + c_P r_P^n$$

where the r_k are the roots of the denominator polynomial $A(z)$ and the c_k are the complex coefficients required for the expansion. Both forms involve linear equations and least squares techniques.

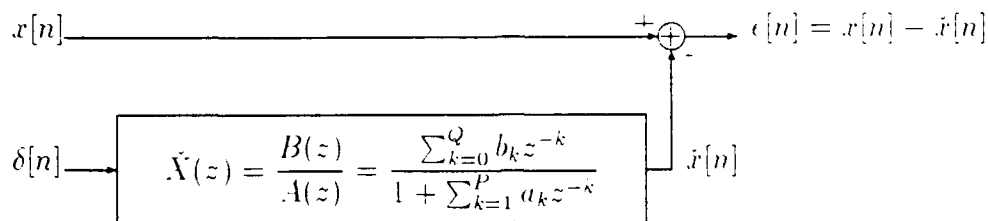


Figure 2.1: Block diagram for the direct method for signal modeling.

The iterative prefiltering method used is due to Steiglitz and McBride [Ref. 7, 8] and attempts to match the data with a model of finite order using an efficient iterative approach. It differs from Prony's method in that it solves for both numerator and denominator polynomial coefficients *simultaneously* at each iteration.

Whenever a signal is modeled using a fixed-order rational polynomial model, an approximation has to be made and some kind of measure has to be used to determine the "goodness" of the model. In this thesis the least squares error norm (*mborl₂* norm) is used to measure the approximation error. This norm measures the energy of the error and is the norm most widely used primarily due to its mathematical tractability [Ref. 2].

B. OVERVIEW OF MODELING METHODS

1. Prony's Method

The derivations of all the modeling methods presented in this section follow those in [Ref. 1, pp. 550-564] and [Ref. 2]. As stated above, Prony's method aims at representing a signal as the impulse response of an LTI system. Figure 2.1 shows an implementation of this approximation which is known as the *direct method*. The system function $\hat{X}(z)$ in the transform domain is a rational polynomial function $B(z)/A(z)$ with Q zeros and P poles. The error signal $e[n]$ is computed as the difference between the response of the system $\hat{x}[n]$ and the given signal $x[n]$, i.e.

$$e[n] = x[n] - \hat{x}[n]. \quad (2.2)$$

The LTI system is chosen to minimize the sum of squared errors

$$\mathcal{S}_D = \sum_n |\epsilon[n]|^2. \quad (2.3)$$

This problem leads to nonlinear equations whose solution (if a unique solution actually exists) turns out to be a very difficult task [Ref. 1, 2]. To avoid these difficulties, a number of indirect methods have been developed for modeling. Prony's method is one of these procedures and can be derived as follows. The LTI system satisfies the difference equation

$$\ddot{x}[n] + a_1 \ddot{x}[n-1] + \cdots + a_P \ddot{x}[n-P] = b_0 \delta[n] + b_1 \delta[n-1] + \cdots + b_Q \delta[n-Q]. \quad (2.4)$$

If the requirement that

$$\ddot{x}[n] = x[n], \quad n = 0, 1, \dots, N_s - 1,$$

is applied to (2.4), where N_s is the length of the data, and the difference equation is evaluated for $n = 0, 1, \dots, N_s - 1$, the result is the matrix equation

$$\begin{bmatrix} x[0] & 0 & 0 & \cdots & 0 \\ x[1] & x[0] & 0 & \cdots & 0 \\ x[2] & x[1] & x[0] & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x[Q] & x[Q-1] & x[Q-2] & \cdots & x[Q-P] \\ x[Q+1] & x[Q] & x[Q-1] & \cdots & x[Q-P+1] \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x[N_s-1] & x[N_s-2] & x[N_s-3] & \cdots & x[N_s-P-1] \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_Q \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (2.5)$$

This can be written as

$$\begin{bmatrix} \mathbf{X}_B \\ \mathbf{X}_A \end{bmatrix} \mathbf{a} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \quad (2.6)$$

where

$$\mathbf{X}_B = \begin{bmatrix} x[0] & 0 & 0 & \cdots & 0 \\ x[1] & x[0] & 0 & \cdots & 0 \\ x[2] & x[1] & x[0] & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x[Q] & x[Q-1] & x[Q-2] & \cdots & x[Q-P] \end{bmatrix} \quad (2.7)$$

and

$$\mathbf{X}_A = \begin{bmatrix} x[Q+1] & x[Q] & x[Q-1] & \cdots & x[Q-P+1] \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x[N_s-1] & x[N_s-2] & x[N_s-3] & \cdots & x[N_s-P-1] \end{bmatrix} \quad (2.8)$$

and \mathbf{a} and \mathbf{b} are the vectors of coefficients appearing in (2.5). The lower partition

$$\mathbf{X}_A \mathbf{a} = \mathbf{0} \quad (2.9)$$

represents an overdetermined set of linear equations that need not have an exact solution. This set of equations can be solved by least squares, where \mathbf{a} is chosen to minimize the least squares norm of the equation error $\mathcal{S}_A = \|\mathbf{e}_A\|^2$ in

$$\mathbf{X}_A \mathbf{a} = \mathbf{e}_A. \quad (2.10)$$

This leads to a set of linear equations called the *normal equations*, which can be written compactly as [Ref. 1, pp.536-537]

$$(\mathbf{X}_A^* \mathbf{X}_A) \mathbf{a} = \begin{bmatrix} \mathcal{S}_A \\ \mathbf{0} \end{bmatrix} \quad (2.11)$$

and can be solved for \mathbf{a} and \mathcal{S}_A . Once the vector \mathbf{a} is known, it can be substituted in the upper partition of (2.6)

$$\mathbf{b} = \mathbf{X}_B \mathbf{a} \quad (2.12)$$

to solve for the vector \mathbf{b} . Although it is referred here simply as Prony's method, this procedure is also known as the modern Prony method or the extended Prony method.

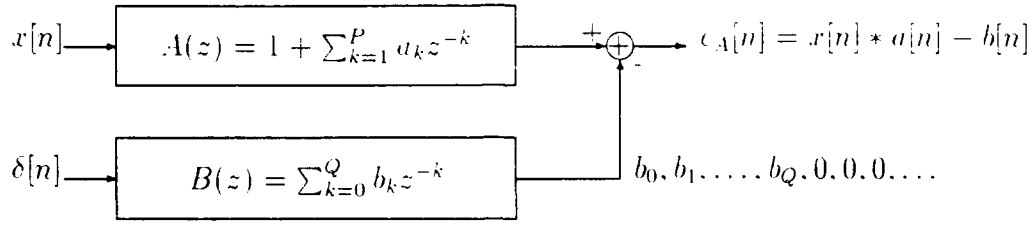


Figure 2.2: Block diagram for the indirect modeling problem.

Although Prony's method is simple to implement, it is important to keep in mind that it is an indirect method. In particular this procedure (see Fig. 2.2) minimizes the squared magnitude of the error

$$\epsilon_A[n] = x[n] * a[n] - b[n] \quad (2.13)$$

where the sequences $a[n]$ and $b[n]$ are defined as

$$a[n] \stackrel{\text{def}}{=} \begin{cases} a_n; & 0 \leq n \leq P \quad (a_0 = 1) \\ 0; & \text{otherwise} \end{cases} \quad (2.14)$$

and

$$b[n] \stackrel{\text{def}}{=} \begin{cases} a_n; & 0 \leq n \leq Q \quad (a_0 = 1) \\ 0; & \text{otherwise} \end{cases} \quad (2.15)$$

where P and Q are the number of poles and zeros respectively. Equation 2.13 represents a different least squares error from that in (2.3) where the quantity to minimize was the squared magnitude of the error $e[n]$ (see Fig. 2.1). The practical significance of this difference is that frequently there is a loss of accuracy in estimating the poles and zeros by Prony's method [Ref. 3].

2. Signal Domain Form of Prony's Method

An alternative formulation of the method described above can be obtained if the problem (2.1) is stated in the signal domain by representing $x[n]$ in terms of a set of complex exponentials:

$$x[n] \approx c_1 r_1^n + c_2 r_2^n + \cdots + c_P r_P^n \quad (2.16)$$

where as stated before, the r_k are the roots of the polynomial $A(z)$, assumed to be distinct, and the complex coefficients c_k provide for the linear combination of the P roots.

This approximation can be initially formulated as in the section above and (2.9) can be solved in the least squares sense for the coefficients of $A(z)$ (ie. the vector \mathbf{a}). The roots r_k of $A(z)$ can then be found and (2.16) can be evaluated for $n = 0, 1, \dots, N_s - 1$ to produce the set of equations

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ r_1 & r_2 & \cdots & r_P \\ r_1^2 & r_2^2 & \cdots & r_P^2 \\ \vdots & \vdots & \cdots & \vdots \\ r_1^{N_s-1} & r_2^{N_s-1} & \cdots & r_P^{N_s-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N_s - 1] \end{bmatrix}. \quad (2.17)$$

This set of equations can then be solved in a least squares sense to obtain the vector of coefficients \mathbf{c} .

In the case of multiple roots at the same location, a slight variation of the same procedure can be used. Suppose, for example, that r_1 is a double root. In this case, the approximation is

$$x[n] \approx c_1 r_1^n + c_2 n r_1^n + \cdots + c_P r_P^n \quad (2.18)$$

and the matrix equation to solve for the coefficients becomes

$$\begin{bmatrix} 1 & 0 & 1 & \cdots & 1 \\ r_1 & r_1 & r_3 & \cdots & r_P \\ r_1^2 & 2r_1^2 & r_3^2 & \cdots & r_P^2 \\ \vdots & \vdots & \cdots & \vdots & \\ r_1^{N_s-1} & (N_s - 1)r_1^{N_s-1} & r_3^{N_s-1} & \cdots & r_P^{N_s-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N_s - 1] \end{bmatrix}. \quad (2.19)$$

This situation is rare, however, because computational errors and errors inherent to the modeling method itself contribute to produce roots that may be very close to each other, but not exactly at the same location.

3. Iterative Prefiltering

The iterative prefiltering method attempts to solve the "direct" problem mentioned in subsection 1 of this chapter and to refine the initial pole-zero estimate by solving a succession of linear problems. Equation 2.2 (error for the direct problem) can be written in the z -domain as

$$E(z) = X(z) - \hat{X}(z) = X(z) - \frac{B(z)}{A(z)} = \frac{X(z)A(z) - B(z)}{A(z)}. \quad (2.20)$$

The notion of iteration can be introduced that allows computation of a new set of poles and zeros based on the last known set of poles. Iterative prefiltering replaces the error for the direct problem at the $(i+1)^{th}$ iteration with the iterative error function

$$E^{(i+1)}(z) = \frac{X(z)A^{(i+1)}(z) - B^{(i+1)}(z)}{A^{(i)}(z)}. \quad (2.21)$$

If $h^{(i)}[n]$ is the inverse z -transform of $1/A^{(i)}(z)$, then the error can be written in the signal domain as

$$e^{(i+1)}[n] = x[n] * h^{(i)}[n] * a^{(i+1)}[n] - b^{(i+1)}[n] * h^{(i)}[n]. \quad (2.22)$$

The coefficients $a^{(i+1)}[n]$ and $b^{(i+1)}[n]$ are selected to minimize the corresponding sum of squared errors

$$\mathcal{S}^{(i+1)} = \sum_{n=P}^{N_S-1} |e^{(i+1)}[n]|^2 \quad (2.23)$$

at each iteration; this situation is shown in Fig. 2.3. No general proof of convergence has been given for this algorithm; however, it is easy to see that if the iteration does converge, it must produce the same answer as the direct method. Specifically, at convergence $A^{(i)} = A^{(i+1)}$, and (2.21) becomes the same as (2.20).

If we use an indirect modeling procedure like Prony's method to compute the initial vector \mathbf{a} and we define $x^{(i)}$ as

$$x^{(i)}[n] \stackrel{\text{def}}{=} x[n] * h^{(i)}[n], \quad (2.24)$$

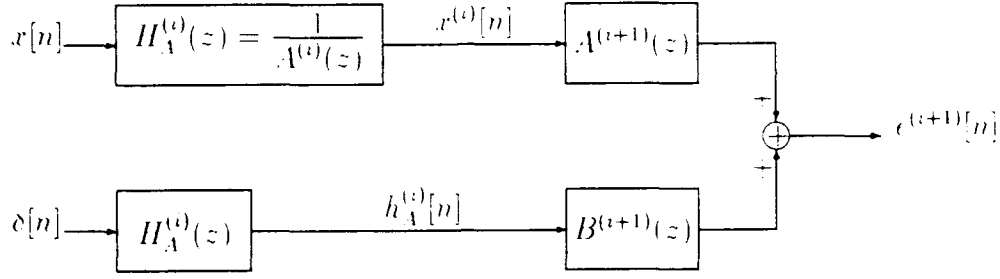


Figure 2.3: Block diagram for the iterative prefiltering method.

then the sequences $h^{(i)}[n]$ and $x^{(i)}[n]$ for $n = 0, 1, \dots, N_S - 1$ can be computed from the recursive difference equations

$$h^{(i)}[n] = d[n] - \sum_{k=1}^P a_k^{(i)} h^{(i)}[n - k] \quad (2.25)$$

and

$$x^{(i)}[n] = x[n] - \sum_{k=1}^P a_k^{(i)} x^{(i)}[n - k]. \quad (2.26)$$

Thus the error (2.22) can be written as

$$\epsilon^{(i+1)}[n] = \sum_{k=0}^P a_k^{(i+1)} x^{(i)}[n - k] - \sum_{j=0}^Q b_j^{(i+1)} h^{(i)}[n - j]. \quad (2.27)$$

In order to find the coefficients $a_k^{(i+1)}$ and $b_j^{(i+1)}$, the error is written in matrix form for $n = 0, 1, \dots, N_S - 1$ as

$$\begin{bmatrix} \mathbf{X}^{(i)} & \mathbf{H}^{(i)} \end{bmatrix} \begin{bmatrix} \mathbf{a}^{(i+1)} \\ -\mathbf{b}^{(i+1)} \end{bmatrix} = \mathbf{e}^{(i+1)}, \quad (2.28)$$

where

$$\mathbf{X}^{(i)} = \begin{bmatrix} x^{(i)}[P] & x^{(i)}[P-1] & \cdots & x^{(i)}[0] \\ x^{(i)}[P+1] & x^{(i)}[P] & \cdots & x^{(i)}[1] \\ \vdots & \vdots & \cdots & \vdots \\ x^{(i)}[N_S-1] & x^{(i)}[N_S-2] & \cdots & x^{(i)}[N_S-1-P] \end{bmatrix} \quad (2.29)$$

$$\mathbf{H}^{(i)} = \begin{bmatrix} h^{(i)}[P] & h^{(i)}[P-1] & \cdots & h^{(i)}[P-Q] \\ h^{(i)}[P+1] & h^{(i)}[P] & \cdots & h^{(i)}[P-Q+1] \\ \vdots & \vdots & \cdots & \vdots \\ h^{(i)}[N_S-1] & h^{(i)}[N_S-2] & \cdots & h^{(i)}[N_S-1-Q] \end{bmatrix} \quad (2.30)$$

and

$$\mathbf{a}^{(i+1)} = \begin{bmatrix} 1 \\ a_1^{(i+1)} \\ \vdots \\ a_p^{(i+1)} \end{bmatrix}, \quad \mathbf{b}^{(i+1)} = \begin{bmatrix} b_0^{(i+1)} \\ b_1^{(i+1)} \\ \vdots \\ b_Q^{(i+1)} \end{bmatrix}. \quad (2.31)$$

Equation 2.28 is analogous to (2.10). Thus the least squares problem defined by minimizing the norm of the error in (2.28) can be reduced to

$$\left(\begin{bmatrix} \mathbf{X}^{(i)} & \mathbf{H}^{(i)} \end{bmatrix} \right)^* \begin{bmatrix} \mathbf{X}^{(i)} & \mathbf{H}^{(i)} \end{bmatrix} \begin{bmatrix} \mathbf{a}^{(i+1)} \\ -\mathbf{b}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \mathcal{S}^{(i+1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.32)$$

where

$$\mathcal{S}^{(i+1)} = \|\mathbf{e}^{(i+1)}\|^2. \quad (2.33)$$

This linear equation can then be solved for the vectors of filter parameters.

At this point, it is instructive to compare the performance of the three modeling methods outlined in this chapter by applying each to model a small segment of a transient sound corresponding to a wrench being struck. This wrench sound was recorded and sampled in the laboratory at a sampling rate of approximately 10,240 Hz. This signal, denoted by wren01, was also used and modeled in [Ref. 9]. Figure 2.4 shows a 100 point segment of the signal wren01 with one of the three models (Prony's, signal domain form of Prony's, and Iterative Prefiltering) overlaid. In all three cases the signal was modeled with four poles and four zeros. As can be seen, the difference between iterative prefiltering and the first two models is significant. The non-iterative methods match only the initial points of the sequence, and produce poor approximations in modeling the remaining part of the signal. This is due, in large part, to poles not sufficiently close to the unit circle. Iterative prefiltering, on the other hand, produces a model which is close to the real sequence along the complete segment.

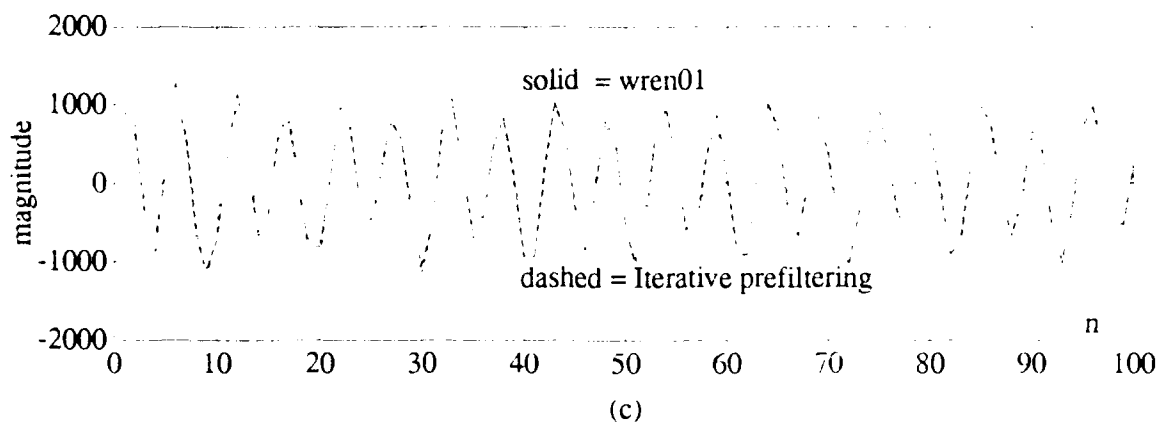
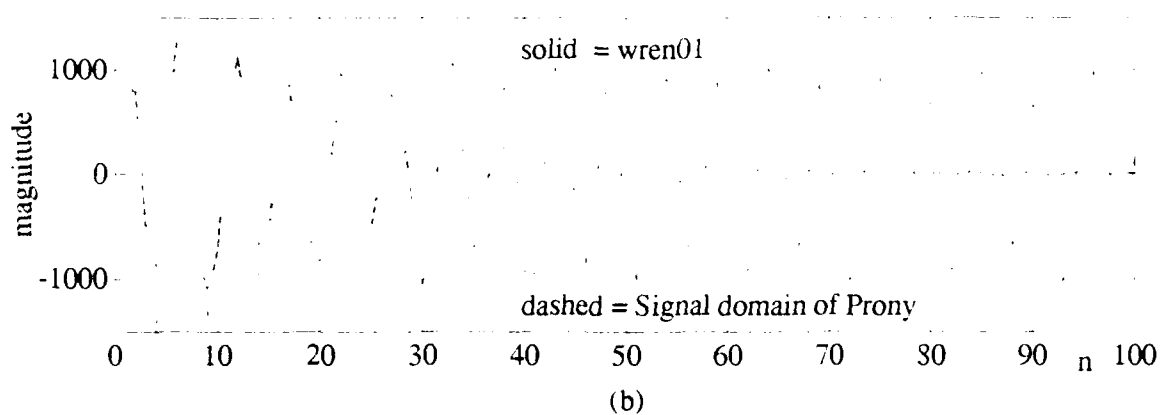
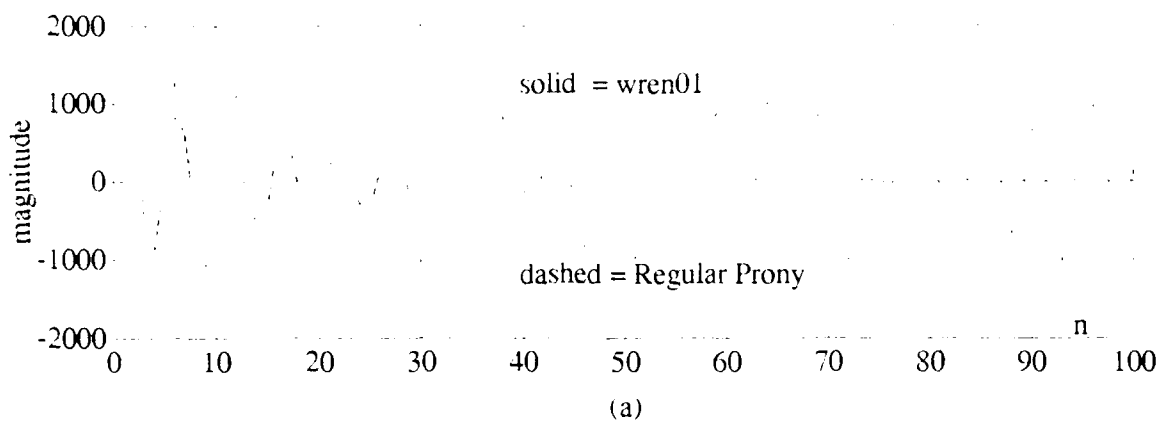


Figure 2.4: Signal *wren01* and its 4 poles/1 zeros models. (a) Using Prony's method. (b) Using Signal Domain form of Prony's method. (c) Using Iterative Prefiltering.

III. ITERATIVE ALGORITHM IN THE SIGNAL DOMAIN

A. MULTIDIMENSIONAL OPTIMIZATION BY GRADIENT METHODS

A multidimensional function $f(x_1, x_2, \dots, x_n)$ that is continuous and differentiable can be minimized using one of several very powerful *hillclimbing* techniques known as *gradient* methods [Ref. 10, p. 84]. Some of those methods are derived on the basis of a quadratic model that can be obtained from a truncated Taylor series expansion of $f(\mathbf{x})$. Let $\mathbf{x}^{(k)}$ denote the value of \mathbf{x} at the k^{th} iteration. Then for any point $\mathbf{x} = \mathbf{x}^{(k)} + \delta$; when δ is small, the function can be approximated by

$$f(\mathbf{x}^{(k)} + \delta) \approx q^{(k)}(\delta) = f^{(k)} + \mathbf{g}^{(k)T} \delta + \frac{1}{2} \delta^T \mathbf{G}^{(k)} \delta \quad (3.1)$$

where \mathbf{g} and \mathbf{G} represent the vector of first derivatives and the matrix of second derivatives of the function $f(\mathbf{x})$ respectively and they should be available at every point. In *Newton's* method the iterate $\mathbf{x}^{(k+1)}$ is taken to be $\mathbf{x}^{(k)} + \delta^{(k)}$, where the correction $\delta^{(k)}$ minimizes $q^{(k)}(\delta)$. This method is only well defined when the matrix of second derivatives \mathbf{G} is positive definite, in which case the k^{th} iteration of *Newton's* method is given by the following procedure [Ref. 11, pp 44-46]:

1. solve $\mathbf{G}^{(k)} \delta = -\mathbf{g}^{(k)}$ for $\delta = \delta^{(k)}$
 2. set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}$.
- (3.2)

The fact that $\mathbf{G}^{(k)}$ may not be positive definite when $\mathbf{x}^{(k)}$ is far from the solution, and that even when $\mathbf{G}^{(k)}$ is positive definite convergence may not occur, makes this method undesirable as a general formulation of a minimization algorithm. However, a

number of variations to the basic method have been proposed that are more suitable for a general class of problems. One of these methods is *Newton's method with line search* [Ref. 11, pp. 47-49] in which the Newton algorithm is used to generate a direction of search

$$\mathbf{s}^{(k)} = -\mathbf{G}^{(k)^{-1}} \mathbf{g}^{(k)} \quad (3.3)$$

which can later be used in a line search algorithm to actually calculate the correction δ . In the cases when $\mathbf{G}^{(k)}$ is not positive definite, the linear search can be made along $\pm \mathbf{s}^{(k)}$ choosing the correct sign to ensure a descent direction. However, some difficulties that arise here (like very high numerical costs and failure of convergence for some special cases) make this an undesirable approach for our algorithm.

As stated before, Newton's method is defined only when the matrix $\mathbf{G}^{(k)}$ is positive definite, and this matrix is positive definite only when the error δ is "small": or better stated, the method is defined only in some neighborhood $\Omega^{(k)}$ of $\mathbf{x}^{(k)}$ in which $q^{(k)}(\delta)$ agrees with $f(\mathbf{x}^{(k)} + \delta)$ in some sense. In such cases, it is correct to choose $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}$, with the correction $\delta^{(k)}$ minimizing $q^{(k)}(\delta)$ for all $\mathbf{x}^{(k)} + \delta$ in $\Omega^{(k)}$. This method is referred to as the *restricted step method* because the step is restricted by the region of validity of the Taylor series. [Ref. 11]

The region of definition for the k^{th} iteration can be expressed as

$$\Omega^{(k)} = \{ \mathbf{x} : \|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \mathbf{h}^{(k)} \} \quad (3.4)$$

where $\|\cdot\|$ denotes the norm of the vector. In this case, the optimization problem can be stated as:

$$\text{minimize}_{\delta} q^{(k)}(\delta) \text{ subject to } \|\delta\| \leq \mathbf{h}^{(k)}. \quad (3.5)$$

As mentioned before, the least squares norm is the one most commonly used in this type of problems, so it is the one used in this thesis and is denoted as $\|\cdot\|_2$. The problem that now becomes apparent is how to select the error margin $\mathbf{h}^{(k)}$ of the

neighborhood (3.4). This margin should be as large as possible because the iteration step is directly related to it. Various methods have been proposed to control the parameter $h^{(k)}$: one of these methods attempts to insure that the Newton's search direction problem (3.3) is always defined [Ref. 11, pp. 100-103]. It does so by adding a multiple of the unit matrix \mathbf{I} to $\mathbf{G}^{(k)}$ and computing the new problem

$$(\mathbf{G}^{(k)} + \nu \mathbf{I}) \delta^{(k)} = -\mathbf{g}^{(k)} \quad (3.6)$$

where the net effect is that increases in ν cause $\|\delta\|_2$ to decrease, and vice versa.

If we define

$$r^{(k)} = \frac{\Delta f^{(k)}}{\Delta q^{(k)}} = \frac{f^{(k)} - f(\mathbf{x}^{(k)} + \delta^{(k)})}{f^{(k)} - q^{(k)}(\delta^{(k)})}, \quad (3.7)$$

then the ratio $r^{(k)}$ represents a measure of the accuracy to which $q^{(k)}(\delta^{(k)})$ approximates $f(\mathbf{x}^{(k)} + \delta^{(k)})$ on the k^{th} step, and as the accuracy increases $r^{(k)}$ gets closer to unity. Using (3.7), Marquardt [Ref. 12] suggests an algorithm that tries to adaptively maintain $h^{(k)}$ as large as possible while controlling the ratio $r^{(k)}$. The k^{th} iteration of such an algorithm is stated as:

1. given $\mathbf{x}^{(k)}$ and $\nu^{(k)}$, calculate $\mathbf{g}^{(k)}$ and $\mathbf{G}^{(k)}$;
2. factor $\mathbf{G}^{(k)} + \nu^{(k)}\mathbf{I}$; if not positive definite, reset $\nu^{(k)} = 4\nu^{(k)}$ and repeat;
3. solve (3.6) to find $\delta^{(k)}$;
4. evaluate $f(\mathbf{x}^{(k)} + \delta^{(k)})$ and hence $r^{(k)}$;
5. if $r^{(k)} < 0.25$ set $\nu^{(k+1)} = 4\nu^{(k)}$
 else if $r^{(k)} > 0.75$ set $\nu^{(k+1)} = \nu^{(k)}/2$
 else set $\nu^{(k+1)} = \nu^{(k)}$;
6. if $r^{(k)} \leq 0$ set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ else set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}$.

Here the parameters 0.25, 0.75, 4 and 2 are arbitrary, and $\nu^{(1)} > 0$ is also chosen arbitrarily [Ref. 11, pp. 102-103]. Proofs of global and second order convergence for this algorithm are given in [Ref. 11, pp. 96-98] for the cases when the first and second derivatives of the function $f(\mathbf{x})$ exist, and the vector $\mathbf{x}^{(k)}$ belongs to a bounded

n -dimensional space for all k . Although this method does have some disadvantages, it represents a good basis for the formulation of a general minimization algorithm. Some variations of this method were considered, but it was found that for the specific application of ARMA modeling in the time domain, these variations produced an extremely high overhead in calculations.

B. THE ITERATIVE PRONY METHOD

Let us now return to the problem of representing a sequence $x[n]$ as a linear combination of complex exponentials. Equation 2.17 can be written as

$$\mathbf{R}\mathbf{c} = \mathbf{x} + \boldsymbol{\epsilon} \quad (3.8)$$

where $\boldsymbol{\epsilon}$ is called the *equation error*, \mathbf{x} represents the data which may or may not be complex, \mathbf{c} is the vector of complex coefficients, and \mathbf{R} is the matrix of complex roots, which can be written more specifically as

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ r_{R_1} + jr_{I_1} & r_{R_2} + jr_{I_2} & \cdots & r_{R_P} + jr_{I_P} \\ (r_{R_1} + jr_{I_1})^2 & (r_{R_2} + jr_{I_2})^2 & \cdots & (r_{R_P} + jr_{I_P})^2 \\ \vdots & \vdots & \cdots & \vdots \\ (r_{R_1} + jr_{I_1})^{N_s-1} & (r_{R_2} + jr_{I_2})^{N_s-1} & \cdots & (r_{R_P} + jr_{I_P})^{N_s-1} \end{bmatrix} \quad (3.9)$$

where r_{R_i} and r_{I_i} represent the real and imaginary components of the i^{th} root, respectively, and N_s is the number of data samples.

By defining

$$Q \stackrel{\text{def}}{=} \|\boldsymbol{\epsilon}\|_2 = \boldsymbol{\epsilon}^{*T} \boldsymbol{\epsilon} = (\mathbf{R}\mathbf{c} - \mathbf{x})^{*T} (\mathbf{R}\mathbf{c} - \mathbf{x}), \quad (3.10)$$

it is clear that the problem is to find the vector $\mathbf{r} = \mathbf{r}_R + j\mathbf{r}_I$ of P complex roots that minimizes the function $Q(\mathbf{r})$.

The first and second derivatives of Q with respect to \mathbf{r} are represented by the

vector \mathbf{g} and matrix \mathbf{G} , which are defined by

$$\mathbf{g} = \begin{bmatrix} \frac{\partial Q}{\partial r_{R_1}} \\ \frac{\partial Q}{\partial r_{R_2}} \\ \vdots \\ \frac{\partial Q}{\partial r_{RP}} \\ \frac{\partial Q}{\partial r_{I_1}} \\ \frac{\partial Q}{\partial r_{I_2}} \\ \vdots \\ \frac{\partial Q}{\partial r_{IP}} \end{bmatrix} \quad (3.11)$$

and

$$\mathbf{G} = \begin{bmatrix} \frac{\partial^2 Q}{\partial r_{R_1} \partial r_{R_1}} & \frac{\partial^2 Q}{\partial r_{R_1} \partial r_{R_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{R_1} \partial r_{RP}} & \frac{\partial^2 Q}{\partial r_{R_1} \partial r_{I_1}} & \frac{\partial^2 Q}{\partial r_{R_1} \partial r_{I_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{R_1} \partial r_{IP}} \\ \frac{\partial^2 Q}{\partial r_{R_2} \partial r_{R_1}} & \frac{\partial^2 Q}{\partial r_{R_2} \partial r_{R_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{R_2} \partial r_{RP}} & \frac{\partial^2 Q}{\partial r_{R_2} \partial r_{I_1}} & \frac{\partial^2 Q}{\partial r_{R_2} \partial r_{I_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{R_2} \partial r_{IP}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 Q}{\partial r_{RP} \partial r_{R_1}} & \frac{\partial^2 Q}{\partial r_{RP} \partial r_{R_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{RP} \partial r_{RP}} & \frac{\partial^2 Q}{\partial r_{RP} \partial r_{I_1}} & \frac{\partial^2 Q}{\partial r_{RP} \partial r_{I_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{RP} \partial r_{IP}} \\ \frac{\partial^2 Q}{\partial r_{I_1} \partial r_{R_1}} & \frac{\partial^2 Q}{\partial r_{I_1} \partial r_{R_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{I_1} \partial r_{RP}} & \frac{\partial^2 Q}{\partial r_{I_1} \partial r_{I_1}} & \frac{\partial^2 Q}{\partial r_{I_1} \partial r_{I_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{I_1} \partial r_{IP}} \\ \frac{\partial^2 Q}{\partial r_{I_2} \partial r_{R_1}} & \frac{\partial^2 Q}{\partial r_{I_2} \partial r_{R_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{I_2} \partial r_{RP}} & \frac{\partial^2 Q}{\partial r_{I_2} \partial r_{I_1}} & \frac{\partial^2 Q}{\partial r_{I_2} \partial r_{I_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{I_2} \partial r_{IP}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 Q}{\partial r_{IP} \partial r_{R_1}} & \frac{\partial^2 Q}{\partial r_{IP} \partial r_{R_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{IP} \partial r_{RP}} & \frac{\partial^2 Q}{\partial r_{IP} \partial r_{I_1}} & \frac{\partial^2 Q}{\partial r_{IP} \partial r_{I_2}} & \cdots & \frac{\partial^2 Q}{\partial r_{IP} \partial r_{IP}} \end{bmatrix}. \quad (3.12)$$

In order to provide for a more compact notation, define the gradient operator with respect to the complex vector \mathbf{r} as the vector of partial derivatives

$$\nabla_{\mathbf{r}} \stackrel{\text{def}}{=} \begin{bmatrix} \nabla_{\mathbf{r}_R} \\ \nabla_{\mathbf{r}_I} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial r_{R_1}} \\ \frac{\partial}{\partial r_{R_2}} \\ \vdots \\ \frac{\partial}{\partial r_{RP}} \\ \frac{\partial}{\partial r_{I_1}} \\ \frac{\partial}{\partial r_{I_2}} \\ \vdots \\ \frac{\partial}{\partial r_{IP}} \end{bmatrix}; \quad (3.13)$$

consequently, (3.11) can be expressed as

$$\mathbf{g} = \nabla_{\mathbf{r}} Q = \begin{bmatrix} \nabla_{\mathbf{r}_R} Q \\ \nabla_{\mathbf{r}_I} Q \end{bmatrix}. \quad (3.14)$$

Equation 3.12 can also be written as

$$\mathbf{G} = \begin{bmatrix} \frac{\partial}{\partial r_{R1}} \left[\begin{array}{cccccc} \frac{\partial Q}{\partial r_{R1}} & \frac{\partial Q}{\partial r_{R2}} & \dots & \frac{\partial Q}{\partial r_{RP}} & \frac{\partial Q}{\partial r_{I1}} & \frac{\partial Q}{\partial r_{I2}} & \dots & \frac{\partial Q}{\partial r_{IP}} \end{array} \right] \\ \frac{\partial}{\partial r_{R2}} \left[\begin{array}{cccccc} \frac{\partial Q}{\partial r_{R1}} & \frac{\partial Q}{\partial r_{R2}} & \dots & \frac{\partial Q}{\partial r_{RP}} & \frac{\partial Q}{\partial r_{I1}} & \frac{\partial Q}{\partial r_{I2}} & \dots & \frac{\partial Q}{\partial r_{IP}} \end{array} \right] \\ \vdots \\ \frac{\partial}{\partial r_{RP}} \left[\begin{array}{cccccc} \frac{\partial Q}{\partial r_{R1}} & \frac{\partial Q}{\partial r_{R2}} & \dots & \frac{\partial Q}{\partial r_{RP}} & \frac{\partial Q}{\partial r_{I1}} & \frac{\partial Q}{\partial r_{I2}} & \dots & \frac{\partial Q}{\partial r_{IP}} \end{array} \right] \\ \frac{\partial}{\partial r_{I1}} \left[\begin{array}{cccccc} \frac{\partial Q}{\partial r_{R1}} & \frac{\partial Q}{\partial r_{R2}} & \dots & \frac{\partial Q}{\partial r_{RP}} & \frac{\partial Q}{\partial r_{I1}} & \frac{\partial Q}{\partial r_{I2}} & \dots & \frac{\partial Q}{\partial r_{IP}} \end{array} \right] \\ \frac{\partial}{\partial r_{I2}} \left[\begin{array}{cccccc} \frac{\partial Q}{\partial r_{R1}} & \frac{\partial Q}{\partial r_{R2}} & \dots & \frac{\partial Q}{\partial r_{RP}} & \frac{\partial Q}{\partial r_{I1}} & \frac{\partial Q}{\partial r_{I2}} & \dots & \frac{\partial Q}{\partial r_{IP}} \end{array} \right] \\ \vdots \\ \frac{\partial}{\partial r_{IP}} \left[\begin{array}{cccccc} \frac{\partial Q}{\partial r_{R1}} & \frac{\partial Q}{\partial r_{R2}} & \dots & \frac{\partial Q}{\partial r_{RP}} & \frac{\partial Q}{\partial r_{I1}} & \frac{\partial Q}{\partial r_{I2}} & \dots & \frac{\partial Q}{\partial r_{IP}} \end{array} \right] \end{bmatrix}. \quad (3.15)$$

Now using a somewhat more convenient notation, (3.15) can be rewritten as

$$\mathbf{G} = \begin{bmatrix} \frac{\partial}{\partial r_{R_i}} \left[\begin{array}{cc} \frac{\partial Q}{\partial r_{R_i}} & \frac{\partial Q}{\partial r_{I_i}} \end{array} \right] \\ \frac{\partial}{\partial r_{I_k}} \left[\begin{array}{cc} \frac{\partial Q}{\partial r_{R_i}} & \frac{\partial Q}{\partial r_{I_i}} \end{array} \right] \end{bmatrix},$$

$$i = 1 \dots P$$

$$k = 1 \dots P \quad (3.16)$$

and it becomes clear that the matrix of second derivatives can be expressed compactly as

$$\mathbf{G} = \nabla_{\mathbf{r}} \left[(\nabla_{\mathbf{r}} Q)^T \right] = \nabla_{\mathbf{r}} \left[(\nabla_{\mathbf{r}_R} Q)^T \quad (\nabla_{\mathbf{r}_I} Q)^T \right]. \quad (3.17)$$

The following subsections derive explicit expressions for the two quantities \mathbf{g} and \mathbf{G} .

1. Vector \mathbf{g} of first derivatives

From (3.8) it is easy to see that

$$\frac{\partial \epsilon}{\partial r_i} = \frac{\partial \mathbf{R}}{\partial r_i} \mathbf{c} \quad \text{and} \quad \frac{\partial \epsilon^{*T}}{\partial r_i} = \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_i} \quad (3.18)$$

where r_i represents the real or imaginary parts of the i^{th} root. Using (3.18) and the chain rule in (3.10) leads to

$$\frac{\partial Q}{\partial r_{R_i}} = \epsilon^{*T} \frac{\partial \mathbf{R}}{\partial r_{R_i}} \mathbf{c} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{R_i}} \epsilon, \quad (3.19)$$

and explicit evaluation of the partial derivatives of the matrix \mathbf{R} results in

$$\begin{aligned} \frac{\partial Q}{\partial r_{R_i}} = & \epsilon^{*T} \begin{bmatrix} 0 \\ 1 \\ 2(r_{R_i} + jr_{I_i}) \\ 3(r_{R_i} + jr_{I_i})^2 \\ \vdots \\ (N_s - 1)(r_{R_i} + jr_{I_i})^{N_s-2} \end{bmatrix} c_i + \\ & c_i^{*T} \begin{bmatrix} 0 & 1 & 2(r_{R_i} - jr_{I_i}) & 3(r_{R_i} - jr_{I_i})^2 & \cdots & (N_s - 1)(r_{R_i} - jr_{I_i})^{N_s-2} \end{bmatrix} \epsilon \end{aligned} \quad (3.20)$$

where c_i is the i^{th} component of the vector \mathbf{c} . If the quantity ξ_i is defined as

$$\xi_i \stackrel{\text{def}}{=} \begin{bmatrix} 0 \\ 1 \\ 2(r_{R_i} + jr_{I_i}) \\ 3(r_{R_i} + jr_{I_i})^2 \\ \vdots \\ (N_s - 1)(r_{R_i} + jr_{I_i})^{N_s-2} \end{bmatrix} c_i, \quad (3.21)$$

then

$$\frac{\partial \mathbf{R}}{\partial r_{R_i}} \mathbf{c} = \xi_i, \quad (3.22)$$

$$\mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{R_i}} = \xi_i^{*T}. \quad (3.23)$$

and (3.20) can be written compactly as

$$\frac{\partial Q}{\partial r_{R_i}} = \epsilon^{*T} \xi_i + \xi_i^{*T} \epsilon. \quad (3.24)$$

At this point it can be recognized that (3.24) represents the addition of two scalar quantities where the first one is the complex conjugate of the second, so (3.24) can be further simplified to

$$\frac{\partial Q}{\partial r_{R_i}} = 2 \operatorname{Re} \left[\xi_i^{*T} \epsilon \right] \quad (3.25)$$

where $\operatorname{Re}[\cdot]$ denotes the real part of the vector. Finally, using (3.25) for $i = 1 \dots P$ in the upper partition of (3.14) produces

$$\nabla_{\mathbf{r}_R} Q = 2 \operatorname{Re} \left\{ \begin{bmatrix} \xi_1^{*T} \\ \xi_2^{*T} \\ \vdots \\ \xi_P^{*T} \end{bmatrix} \epsilon \right\}. \quad (3.26)$$

A similar procedure can be used to obtain the gradient of Q with respect to the imaginary part of the vector \mathbf{r} . Once more, from (3.18) and the chain rule applied to (3.10), it follows that

$$\frac{\partial \mathbf{R}}{\partial r_{I_i}} \mathbf{c} = j \xi_i \quad (3.27)$$

$$\mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{I_i}} = -j \xi_i^{*T}. \quad (3.28)$$

These results can be used to generate an expression similar to that in (3.24) for the vector of first derivatives of Q with respect to the imaginary components of \mathbf{r}

$$\frac{\partial Q}{\partial r_{I_i}} = j \epsilon^{*T} \xi_i - j \xi_i^{*T} \epsilon \quad (3.29)$$

which in turn defines the vector of partial derivatives with respect to the imaginary components as

$$\nabla_{\mathbf{r}_I} Q = 2 \operatorname{Im} \left\{ \begin{bmatrix} \xi_1^{*T} \\ \xi_2^{*T} \\ \vdots \\ \xi_P^{*T} \end{bmatrix} \epsilon \right\}. \quad (3.30)$$

Equations 3.26 and 3.30 can now be combined as shown in (3.14) to obtain the final expression for the vector of first derivatives

$$\mathbf{g} = \begin{bmatrix} \nabla_{\mathbf{r}_R} Q \\ \nabla_{\mathbf{r}_I} Q \end{bmatrix} = 2 \begin{bmatrix} \text{Re} \left\{ \begin{bmatrix} \xi_1^{*T} \\ \xi_2^{*T} \\ \vdots \\ \xi_P^{*T} \end{bmatrix} \epsilon \right\} \\ \text{Im} \left\{ \begin{bmatrix} \xi_1^{*T} \\ \xi_2^{*T} \\ \vdots \\ \xi_P^{*T} \end{bmatrix} \epsilon \right\} \end{bmatrix}. \quad (3.31)$$

2. Matrix G of second derivatives

An expression for the matrix \mathbf{G} of second derivatives can be obtained as follows. Substituting (3.24) and (3.29) into (3.16) yields

$$\mathbf{G} = \begin{bmatrix} \frac{\partial}{\partial r_{R_k}} \left(\begin{bmatrix} \epsilon^{*T} \xi_i + \xi_i^{*T} \epsilon \end{bmatrix} & j \begin{bmatrix} \epsilon^{*T} \xi_i - \xi_i^{*T} \epsilon \end{bmatrix} \end{bmatrix} \\ \frac{\partial}{\partial r_{I_k}} \left(\begin{bmatrix} \epsilon^{*T} \xi_i + \xi_i^{*T} \epsilon \end{bmatrix} & j \begin{bmatrix} \epsilon^{*T} \xi_i - \xi_i^{*T} \epsilon \end{bmatrix} \end{bmatrix} \\ i = 1 \dots P \\ k = 1 \dots P. \quad (3.32)$$

Then using the chain rule and both expressions in (3.18) leads to

$$\mathbf{G} = \begin{bmatrix} \left[\epsilon^{*T} \frac{\partial \xi_i}{\partial r_{R_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{R_k}} \xi_i + \xi_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{R_k}} \mathbf{c} + \frac{\partial \xi_i^{*T}}{\partial r_{R_k}} \epsilon \right] & j \left[\epsilon^{*T} \frac{\partial \xi_i}{\partial r_{R_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{R_k}} \xi_i - \xi_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{R_k}} \mathbf{c} - \frac{\partial \xi_i^{*T}}{\partial r_{R_k}} \epsilon \right] \\ \left[\epsilon^{*T} \frac{\partial \xi_i}{\partial r_{I_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{I_k}} \xi_i + \xi_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{I_k}} \mathbf{c} + \frac{\partial \xi_i^{*T}}{\partial r_{I_k}} \epsilon \right] & j \left[\epsilon^{*T} \frac{\partial \xi_i}{\partial r_{I_k}} + \mathbf{c}^{*T} \frac{\partial \mathbf{R}^{*T}}{\partial r_{I_k}} \xi_i - \xi_i^{*T} \frac{\partial \mathbf{R}}{\partial r_{I_k}} \mathbf{c} - \frac{\partial \xi_i^{*T}}{\partial r_{I_k}} \epsilon \right] \end{bmatrix} \\ i = 1 \dots P; \quad k = 1 \dots P. \quad (3.33)$$

From the definition of ξ in (3.21) it is seen that

$$\frac{\partial \xi_i}{\partial r_{R_k}} = s_{ik} \stackrel{\text{def}}{=} \begin{cases} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 6(r_{R_i} + jr_{I_i}) \\ 12(r_{R_i} + jr_{I_i})^2 \\ \vdots \\ (N_s - 1)(N_s - 2)(r_{R_i} + jr_{I_i})^{N_s-3} \end{bmatrix} c_i, & \text{if } i=k \\ 0, & \text{otherwise.} \end{cases} \quad (3.31)$$

In the same way it can be shown that

$$\frac{\partial \xi_i^{*T}}{\partial r_{R_k}} = s_{ik}^{*T} \quad (3.35)$$

$$\frac{\partial \xi_i}{\partial r_{I_k}} = js_{ik} \quad (3.36)$$

$$\frac{\partial \xi_i^{*T}}{\partial r_{I_k}} = -js_{ik}^{*T}. \quad (3.37)$$

These last four expressions together with (3.22), (3.23), (3.27), and (3.28) can now be substituted in (3.33) to obtain

$$\mathbf{G} = \begin{bmatrix} \left[\epsilon^{*T} s_{ik} + \xi_k^{*T} \xi_i + \xi_i^{*T} \xi_k + s_{ik}^{*T} \epsilon \right] & j \left[\epsilon^{*T} s_{ik} + \xi_k^{*T} \xi_i - \xi_i^{*T} \xi_k - s_{ik}^{*T} \epsilon \right] \\ j \left[\epsilon^{*T} s_{ik} - \xi_k^{*T} \xi_i + \xi_i^{*T} \xi_k - s_{ik}^{*T} \epsilon \right] & - \left[\epsilon^{*T} s_{ik} - \xi_k^{*T} \xi_i - \xi_i^{*T} \xi_k + s_{ik}^{*T} \epsilon \right] \end{bmatrix}$$

$i = 1 \dots P$

$k = 1 \dots P.$ (3.38)

Finally notice again that the elements of the matrix \mathbf{G} are formed by additions and subtractions of complex scalars with their respective complex conjugates:

thus the final expression for \mathbf{G} is given by

$$\mathbf{G} = \begin{bmatrix} 2 \operatorname{Re} [\xi_i^{*T} \xi_k] + 2 \operatorname{Re} [\mathbf{s}_{ik}^{*T} \epsilon] & 2 \operatorname{Im} [\xi_i^{*T} \xi_k] - 2 \operatorname{Im} [\mathbf{s}_{ik}^{*T} \epsilon] \\ -2 \operatorname{Im} [\xi_i^{*T} \xi_k] + 2 \operatorname{Im} [\mathbf{s}_{ik}^{*T} \epsilon] & 2 \operatorname{Re} [\xi_i^{*T} \xi_k] - 2 \operatorname{Re} [\mathbf{s}_{ik}^{*T} \epsilon] \end{bmatrix}$$

$$i = 1 \dots P; \quad j = 1 \dots P \quad (3.39)$$

where it should be remembered that $\mathbf{s}_{ik} = \mathbf{0}$ for all $i \neq k$.

3. Algorithm implementation

An iterative method for ARMA modeling in the time domain was implemented using the results of the last two subsections in conjunction with the algorithm presented in section A of this chapter. We call this method the *iterative Prony* method.

The algorithm uses the signal domain form of Prony's method to calculate an initial model for the given sequence. From there it uses the calculated model to compute the error ϵ , the vector of first derivatives \mathbf{g} , and the matrix of second derivatives \mathbf{G} and iterates until specific conditions are met. Figure 3.1 is an example of how the algorithm changes the position of the poles and zeros of the initial model in order to minimize the error. This figure represents the poles and zeros of a transfer function of order (4,3)—4 poles 3 zeros—that was overmodeled using a (6,5) order model. It is clear from the figure that the tendency in this case is to have a pole-zero cancellation (see second and third quadrants) of two poles and two zeros as expected.

Some features were added to the basic algorithm in order to deal with special modeling cases. Specifically, if the initial model has some roots on the real axis, then because of the way the algorithm iterates, those roots never move away from the real axis. A modification was therefore introduced to deliberately displace those roots from the real axis and proceed with the iterations. If the tendency of the roots is

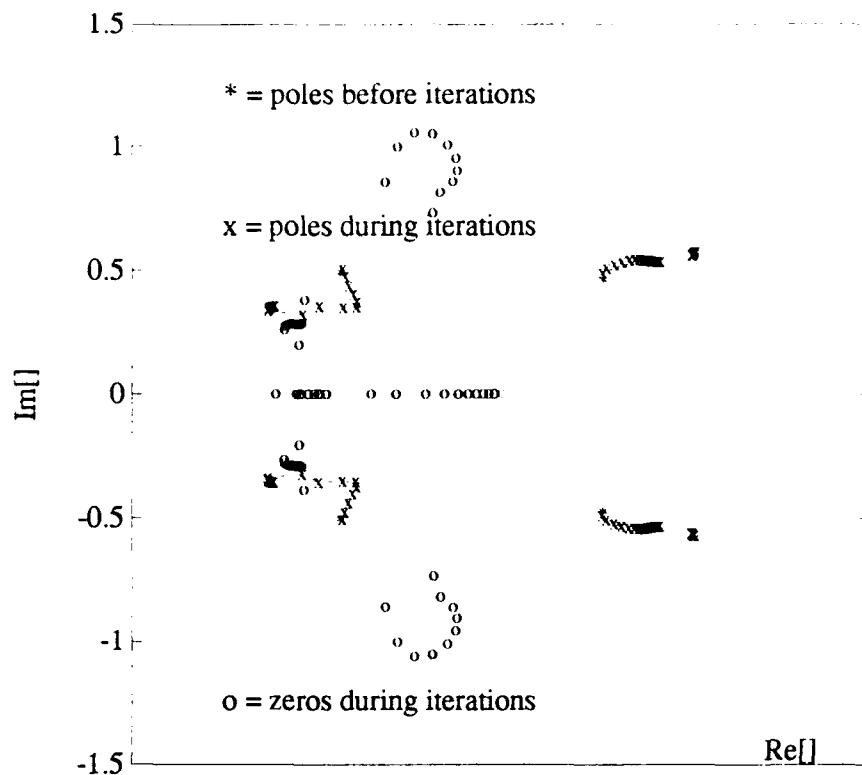


Figure 3.1: Displacement of the poles and zeros of an iterative Prony's model

“to go back” to the real axis, then they are returned to their initial position, and the iterations continue. Otherwise the roots may continue to spread apart and move as a complex pair. Figure 3.2 is an example of the displacement of the poles and zeros of an order (4,3) model. In this case the modeled signal actually has two poles on the real axis, and the initial model correctly placed two of the poles in the real axis. Those poles are displaced from the real axis by the algorithm, but then after some iterations it is clear that the poles are tending to return to the real axis. At this point the poles are forced back to the real axis by setting their imaginary parts to zero and the iterations continue until convergence is obtained. The opposite situation is shown in Figure 3.3. In this case the initial model also has two poles located on the real axis, but contrary to the case presented above, the roots, after being displaced from the real axis, continue to move away from the axis until they reach their final position in the first and fourth quadrants closer to the unit circle.

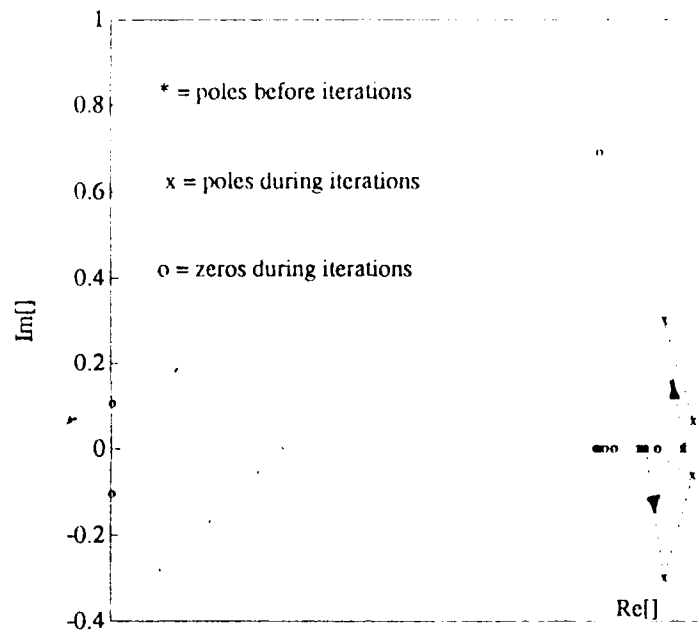


Figure 3.2: Displacement of the poles and zeros of a 4-3 model. The modeled signal in this case actually has two poles on the real axis

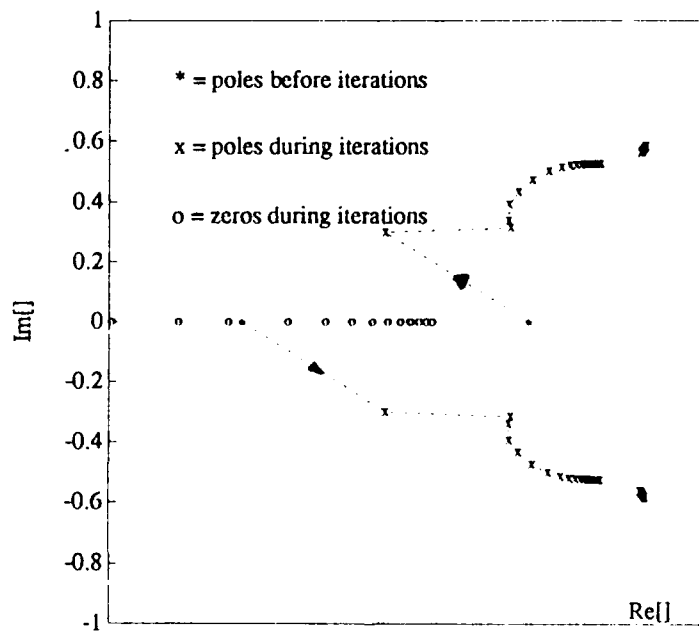


Figure 3.3: Displacement of the poles and zeros of a 4-3 model. The initial model shows two poles in the real axis, the final model in this case has no poles in the axis

IV. PERFORMANCE OF THE ALGORITHM AND MODELING RESULTS

A. TEST DATA USED IN THIS THESIS

Two types of signals were used in this thesis to test the performance of the algorithm. The first type, which will be called *simulated* test data, consists of five sequences (t01 to t05) each one hundred points long that were produced as the impulse response of a known rational system. Noise to produce an SNR in the range of 10 to 15 dB was added to the original sequences, and the resulting sequences were designated as t01_n to t05_n. The original signals are described in Table 4.1 by their transfer functions and the location of their poles and zeros.

The second group of test signals consists of recorded *acoustic* data. Two of these signals were recorded and sampled in the laboratory. One of them is the sequence *wren01* already mentioned in Chapter II; the other one was obtained from human speech, in particular, the signal *vowel_a* corresponds to 100 samples of the spanish vowel *a*. The remaining three signals from the group of *acoustic* data were recorded at sea by a submarine platform: they correspond to sounds produced by marine life and ice cracking. The description of the *acoustic* signals is presented in Table 4.2.

B. PRESENTATION OF RESULTS

All simulated test signals were modeled twice with the iterative Prony method. In the first test the exact number of poles and zeros of the original model was used; in the second test all signals were modeled using two more poles and zeros than the original model. This last test is considered closer to a real life situation where the exact order of the signal to be modeled is unknown.

TABLE 4.1: DESCRIPTION OF *SIMULATED* TEST DATA

NAME	TRANSFER FUNCTION	POLES	ZEROS
t01	$H(z) = \frac{1-0.77z^{-1}}{1-1.539z^{-1}+0.905z^{-2}}$	$0.9513 \angle \pm 0.6286$	0; 0.770
t02	$H(z) = \frac{0.744z^{-1}-1.6993z^{-2}+0.630z^{-3}}{1-2.977z^{-1}+3.909z^{-2}-2.520z^{-3}+0.717z^{-4}}$	$0.9422 \angle \pm 0.6200$ $0.8987 \angle \pm 0.6385$	0; 0.4686 1.8069; ∞
t03	$H(z) = \frac{0.171z^{-1}-0.155z^{-3}}{1+1.861z^{-1}+2.588z^{-2}+1.684z^{-3}+0.819z^{-4}}$	$0.9512 \angle \pm 1.8846$ $0.9514 \angle \pm 2.3041$	0; 0.9521 -0.9521; ∞
t04	$H(z) = \frac{0.981z^{-1}-1.279z^{-2}+0.871z^{-3}}{1-0.956z^{-1}+0.040z^{-2}-0.705z^{-3}+0.741z^{-4}}$	$0.9513 \angle \pm 0.2097$ $0.9049 \angle \pm 2.0943$	0; ∞ $0.9123 \angle \pm 0.8068$
t05	$H(z) = \frac{1-0.75z^{-1}}{1-1.78z^{-1}+0.79z^{-2}}$	0.8441; 0.9358	0; 0.750

TABLE 4.2: DESCRIPTION OF *ACOUSTIC* TEST DATA

SIGNAL	DESCRIPTION
wren01	Transient sound corresponding to a wrench being struck
vowel_a	Spanish vowel <i>a</i>
bio_2133a	Sperm whale
bio_2385a	Porpoise whistle
bio_80a	Ice cracking

To mathematically produce a meaningful measure of the performance of a modeling algorithm can be quite difficult since various norms can be deceiving when comparing errors of signals with large differences in magnitude. Two different approaches to measure the performance of the algorithms were therefore used in this thesis. The first is quantitative and involves computing the squared-norm of the error between the model and the actual signal and dividing it by the total energy (norm) of the signal. The second approach is to overlay in a plot the model and the original signal in order to provide a visual comparison of the results. This is less quantitative but frequently more revealing of errors in the modeling process.

1. Simulated test data

The first data sets modeled were the simulated test data sets. Figure 4.1(a) is a comparison of the normalized errors that result when the sequence $t01_n$ is modeled with 2 poles and 1 zero using both iterative prefiltering and iterative Prony methods. Figure 4.1(b) and (c) show 100 points of the sequence $t01_n$ and the two order (2,1) models. At this point there is no noticeable difference between the iterative prefiltering and the iterative Prony models. Figure 4.2(a) again shows a comparison of the normalized errors between an iterative prefiltering model and an iterative Prony model of $t01_n$ for the case when the signal $t01_n$ was overmodeled using models of order (4,3). Although the difference between the two modeled signals in this case is not large, notice that the error for the iterative prefiltering method initially increases before decreasing while the error for the iterative Prony method decreases monotonically. This is the first example of a pattern that repeats in all but one of the simulated test signals that were modeled. Figures 4.3 through 4.10 give similar comparisons for the remaining simulated signals. The pattern, which can be seen in Figures 4.1 to 4.10, is that when the signals are modeled with a number of poles and zeros different from that of the actual order of the signal (always overmodeling

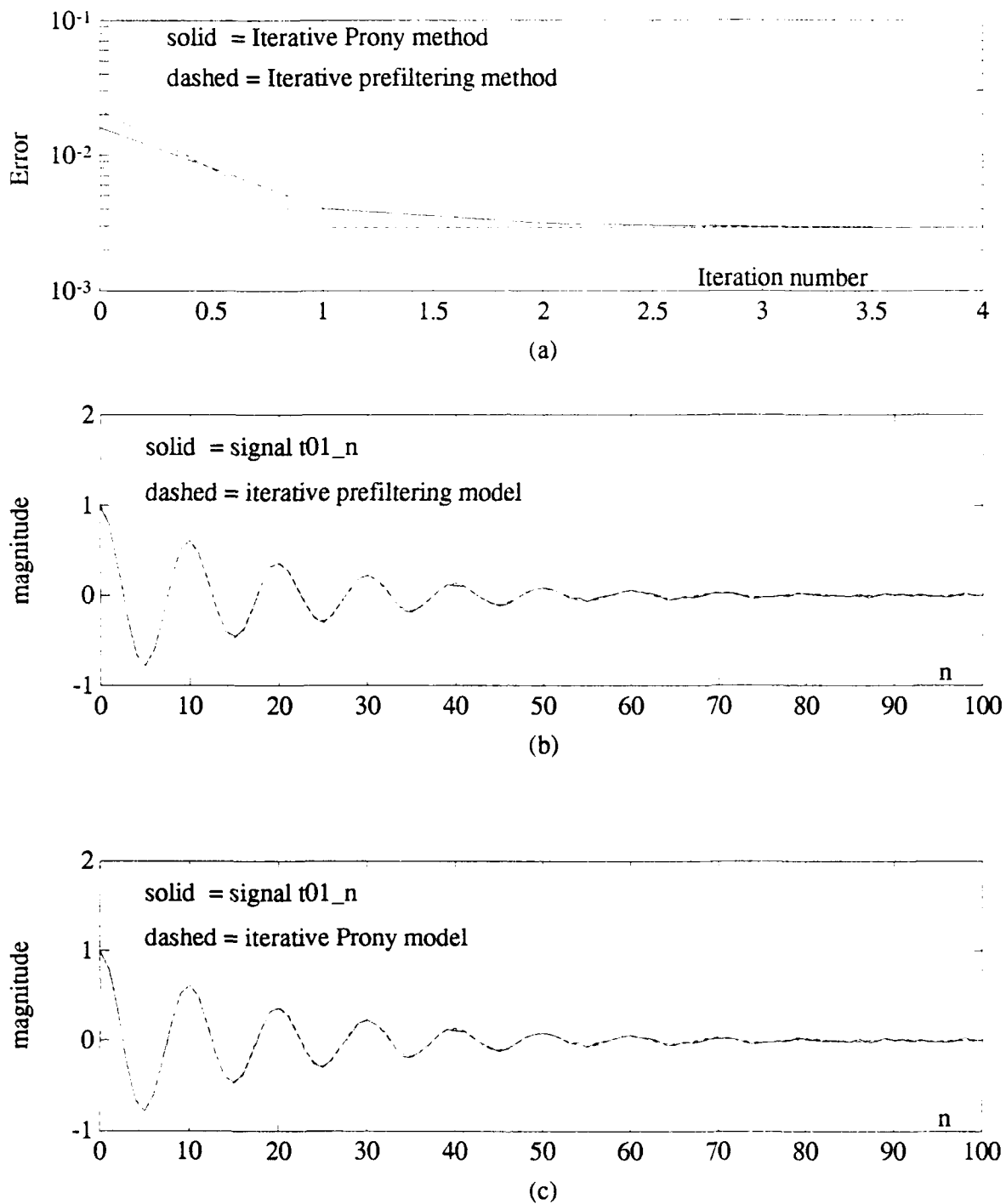


Figure 4.1: Signal $t01_n$ and its **2 poles-1 zero** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t01_n$ and the iterative prefiltering model. (c) Signal $t01_n$ and the iterative Prony model.

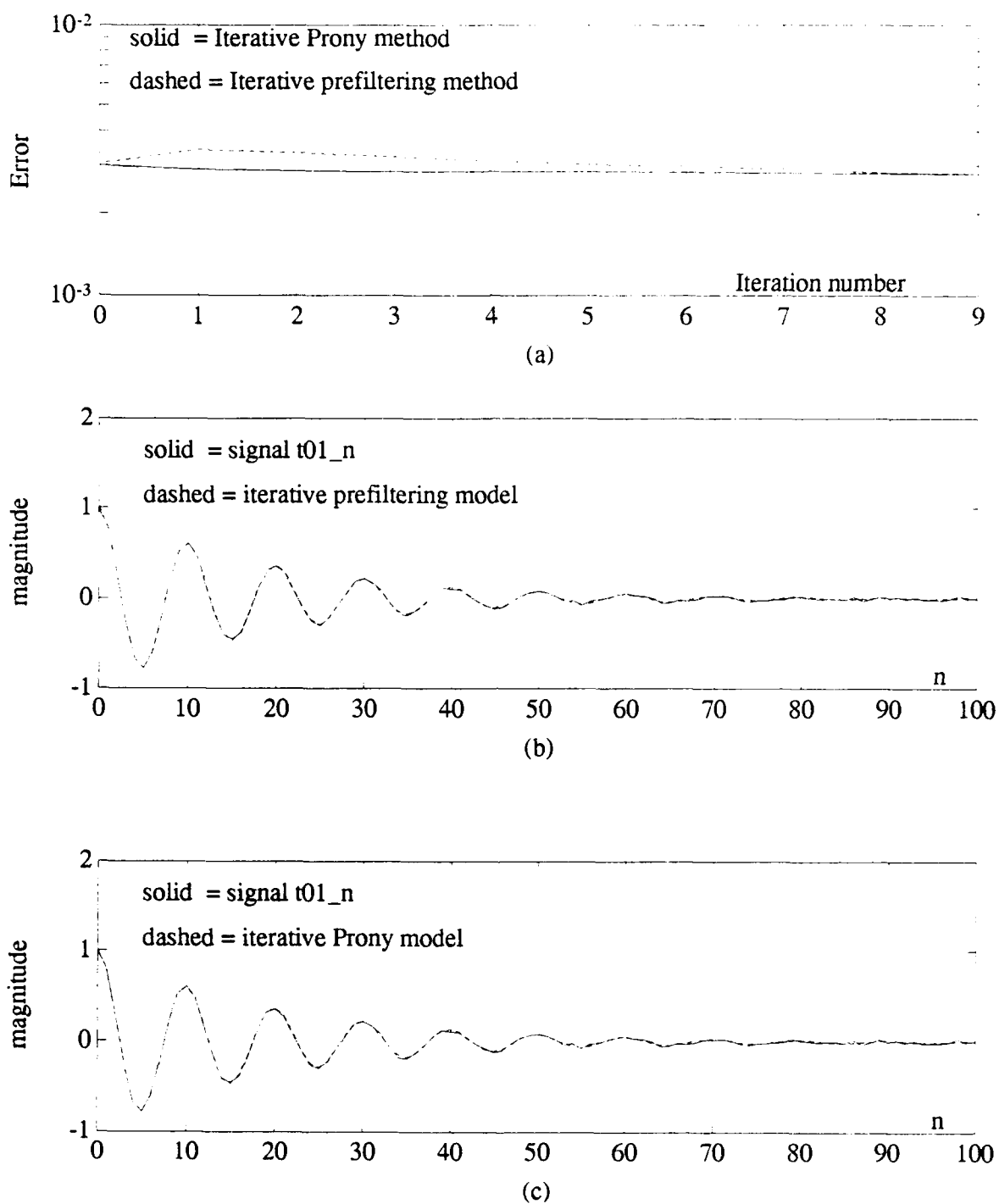


Figure 4.2: Signal $t01_n$ and its **4 poles-3 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t01_n$ and the iterative prefiltering model. (c) Signal $t01_n$ and the iterative Prony model.

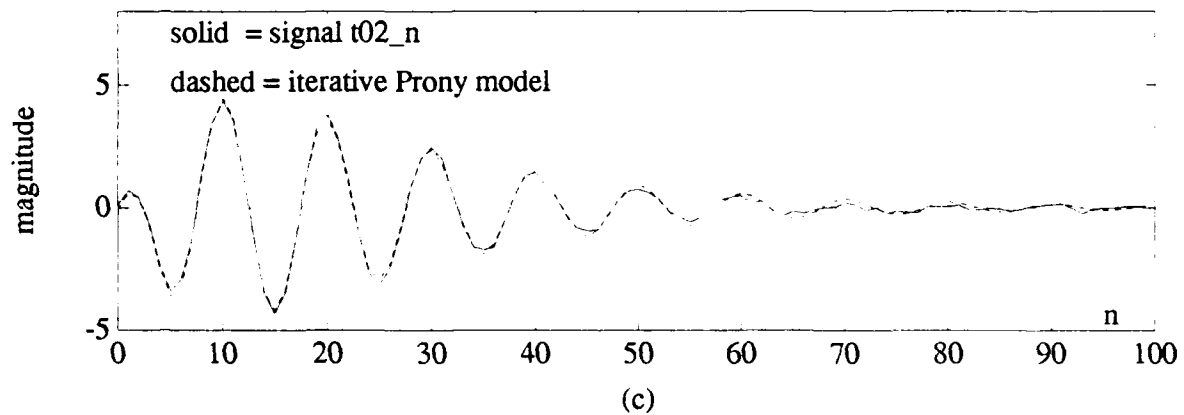
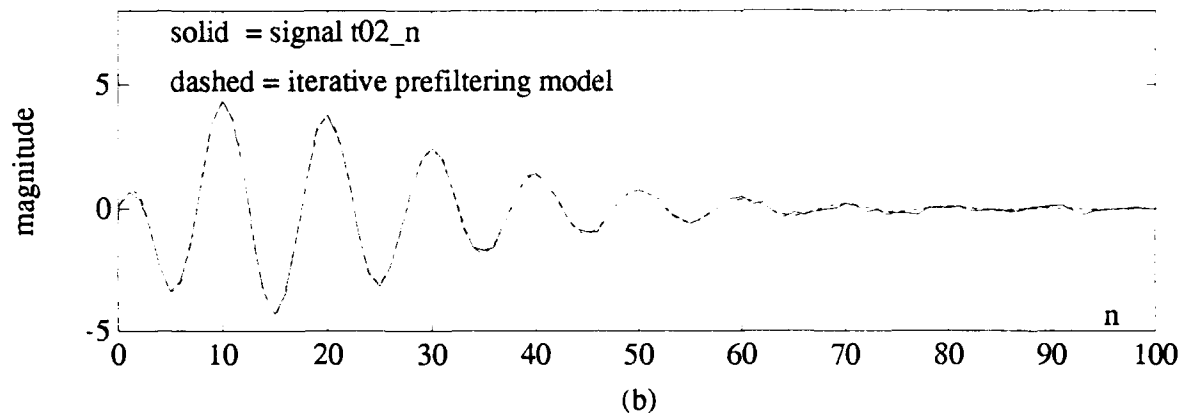
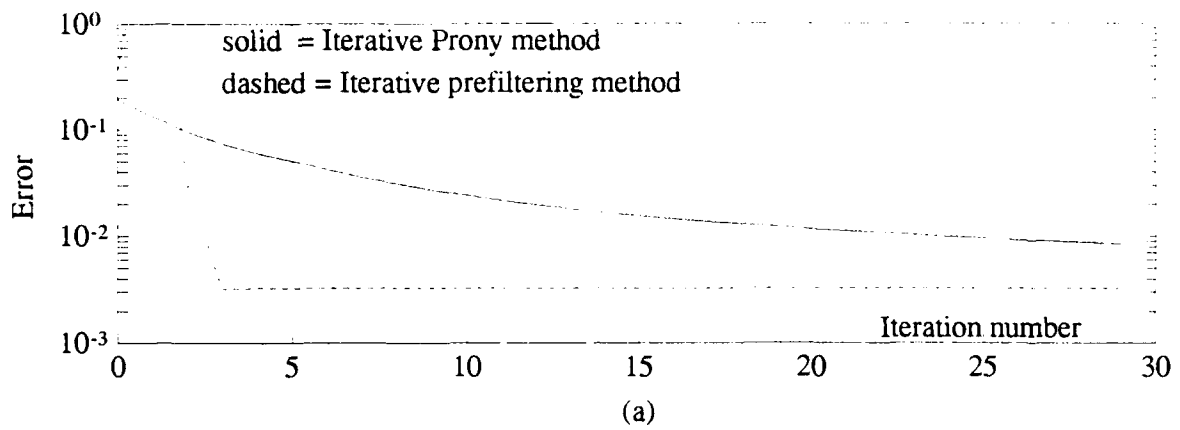


Figure 4.3: Signal $t02_n$ and its **4 poles-3 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t02_n$ and the iterative prefiltering model. (c) Signal $t02_n$ and the iterative Prony model.

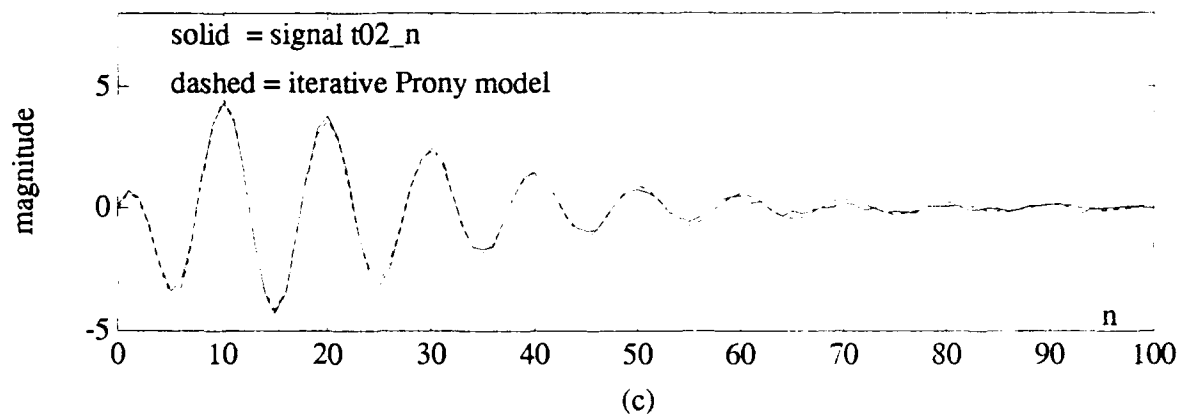
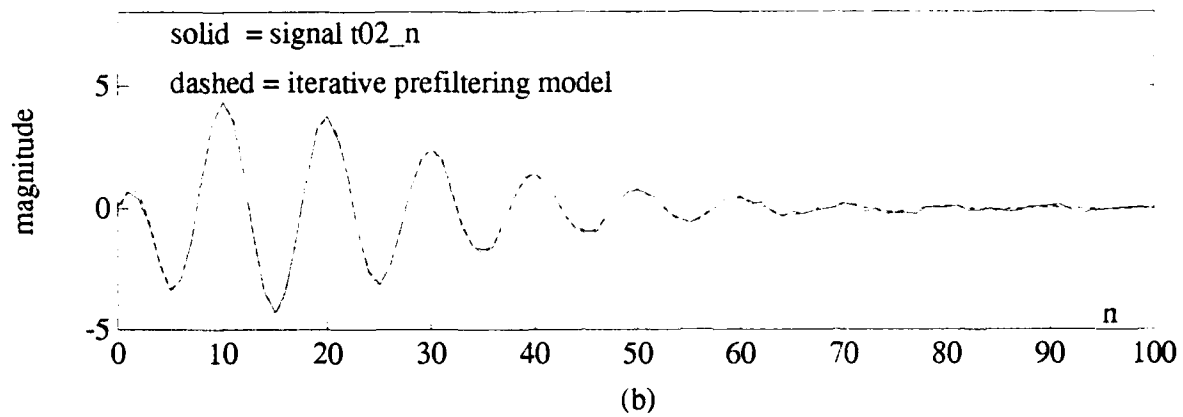
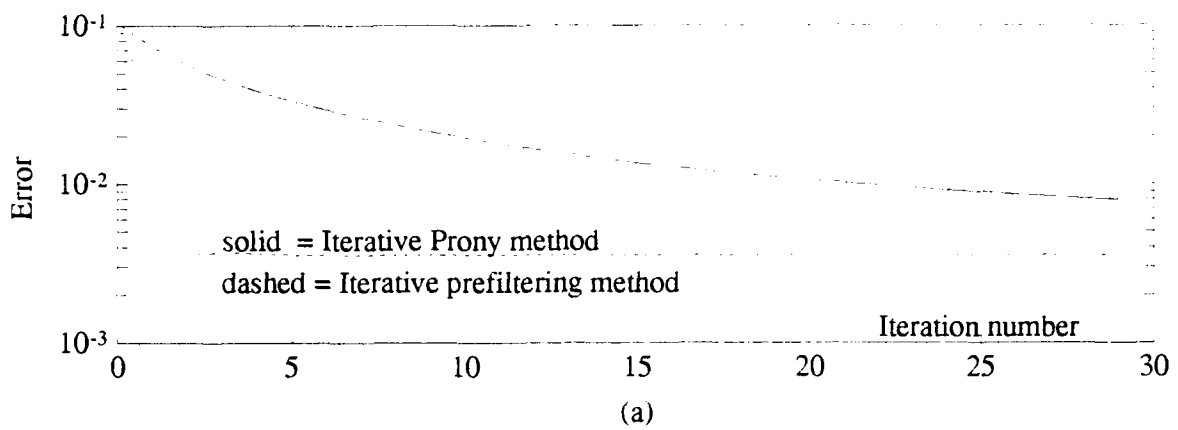


Figure 4.4: Signal $t02_n$ and its **6 poles-5 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t02_n$ and the iterative prefiltering model. (c) Signal $t02_n$ and the iterative Prony model.

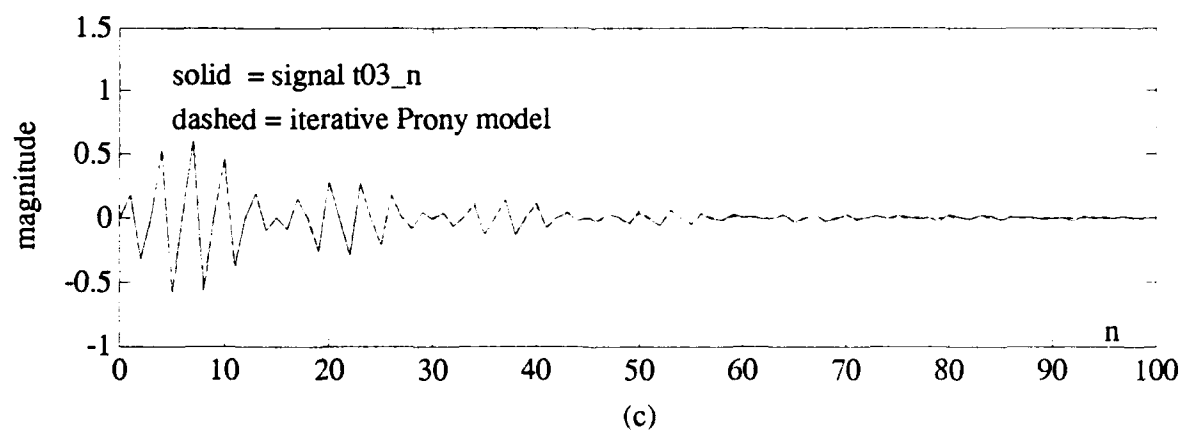
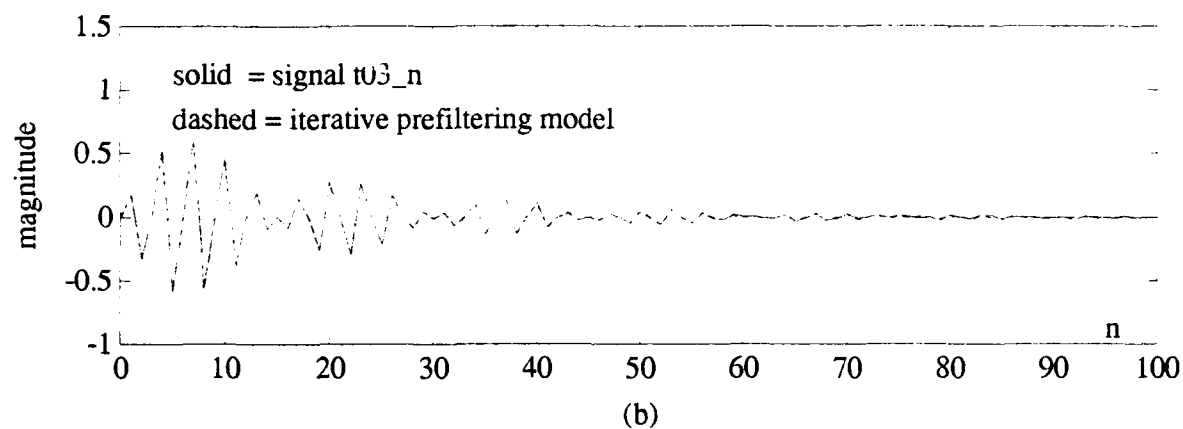
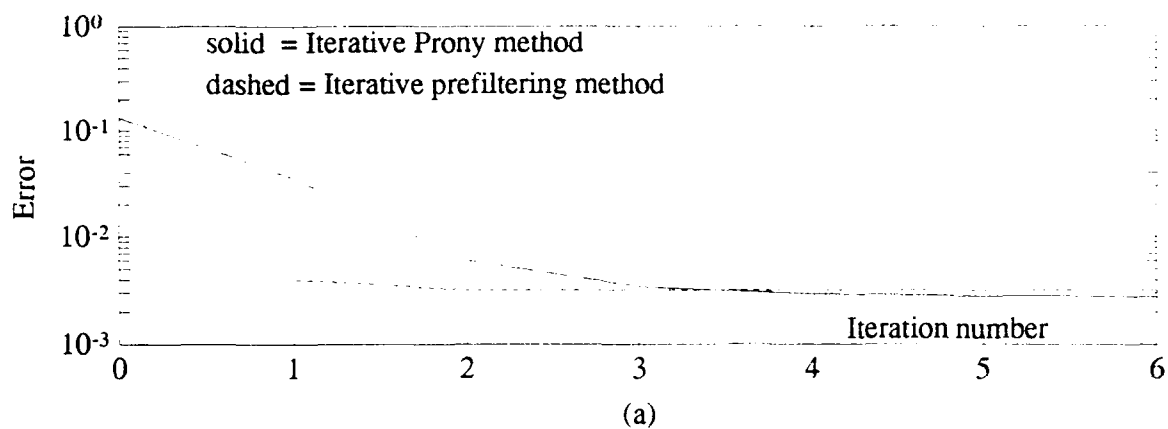


Figure 4.5: Signal $t03_n$ and its **4 poles-3 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t03_n$ and the iterative prefiltering model. (c) Signal $t03_n$ and the iterative Prony model.

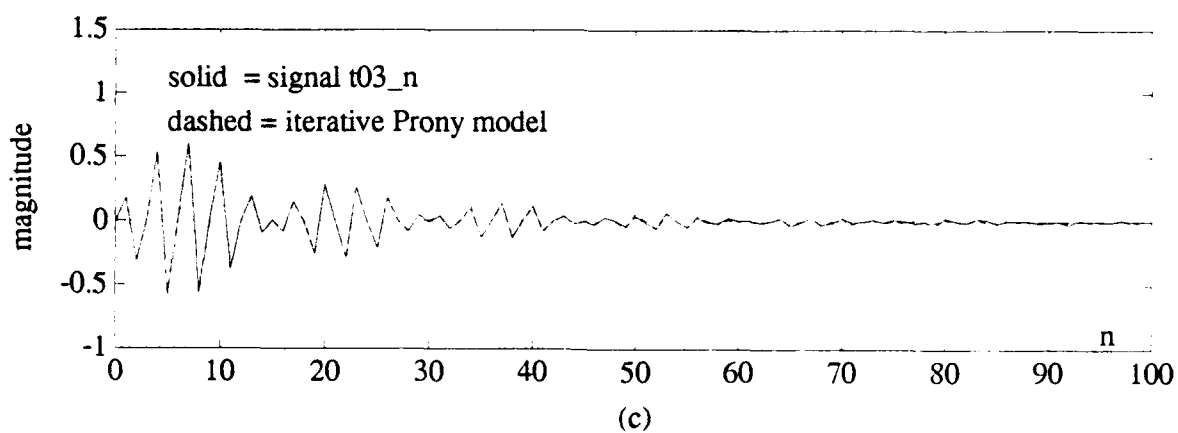
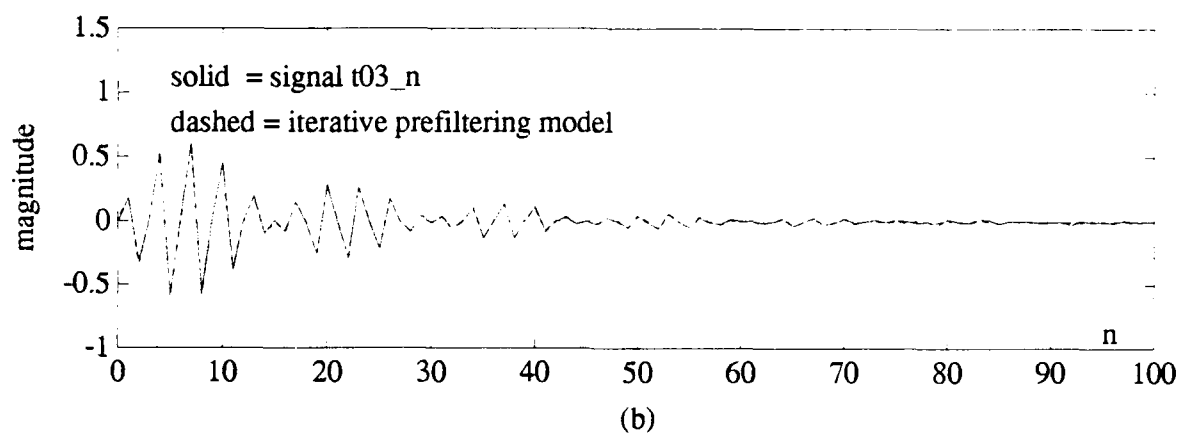
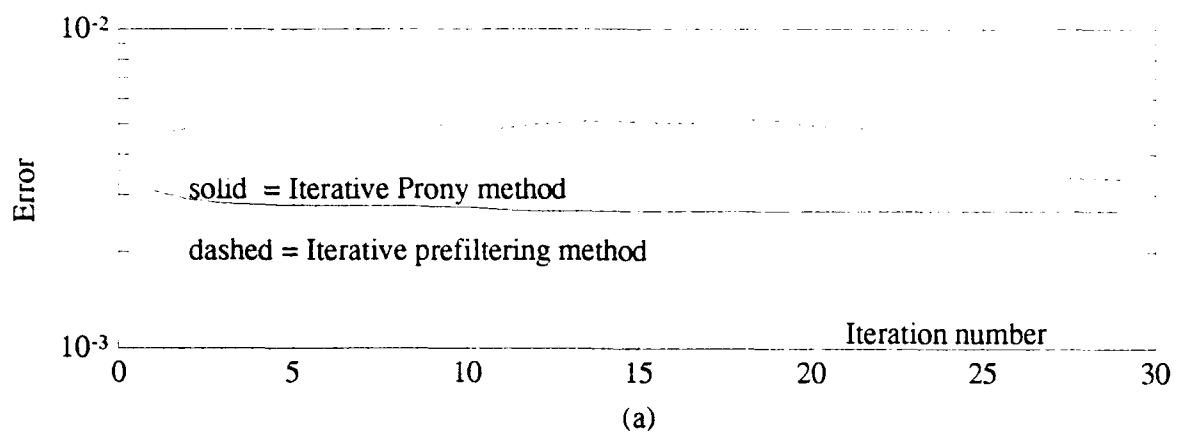


Figure 4.6: Signal $t03_n$ and its **6 poles-5 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t03_n$ and the iterative prefiltering model. (c) Signal $t03_n$ and the iterative Prony model.

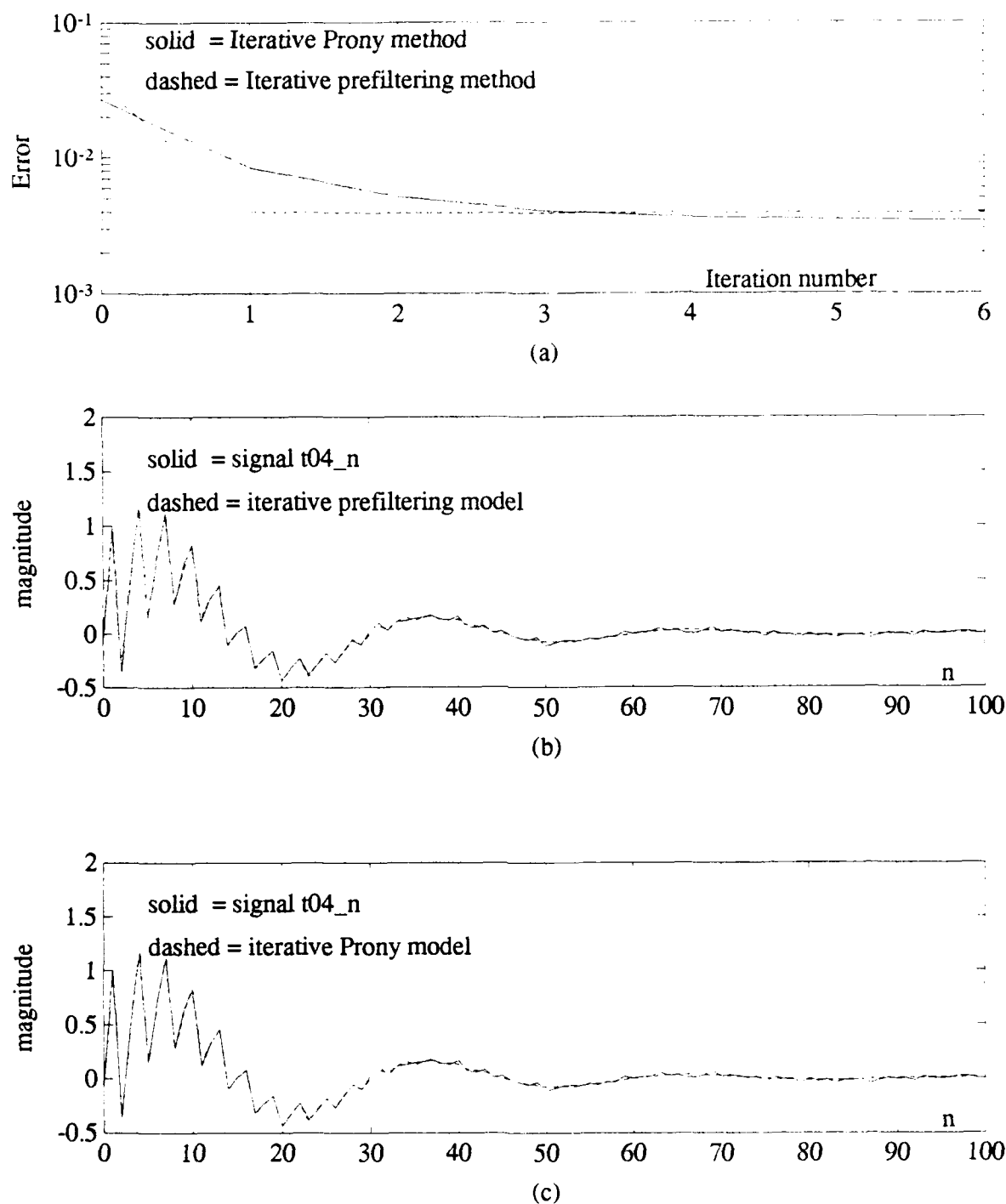


Figure 4.7: Signal $t04_n$ and its **4 poles-3 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t04_n$ and the iterative prefiltering model. (c) Signal $t04_n$ and the iterative Prony model.

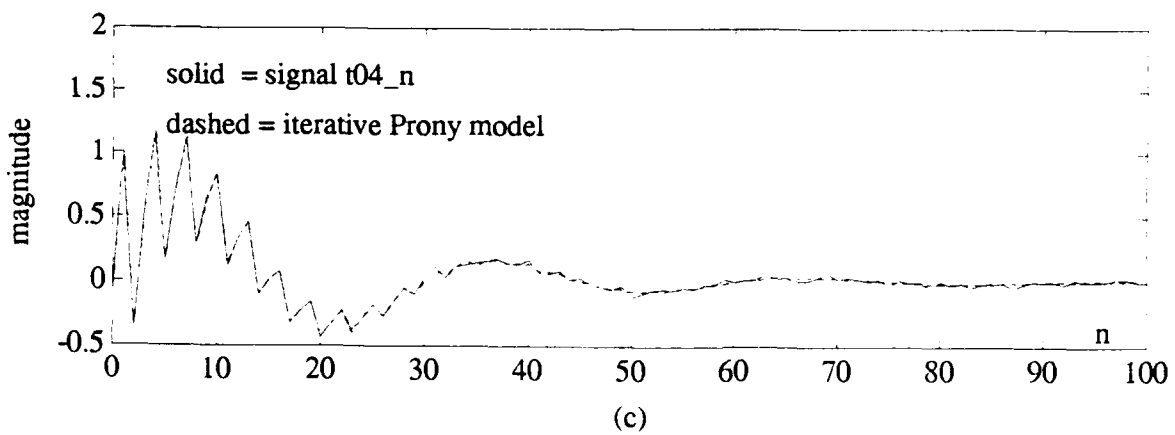
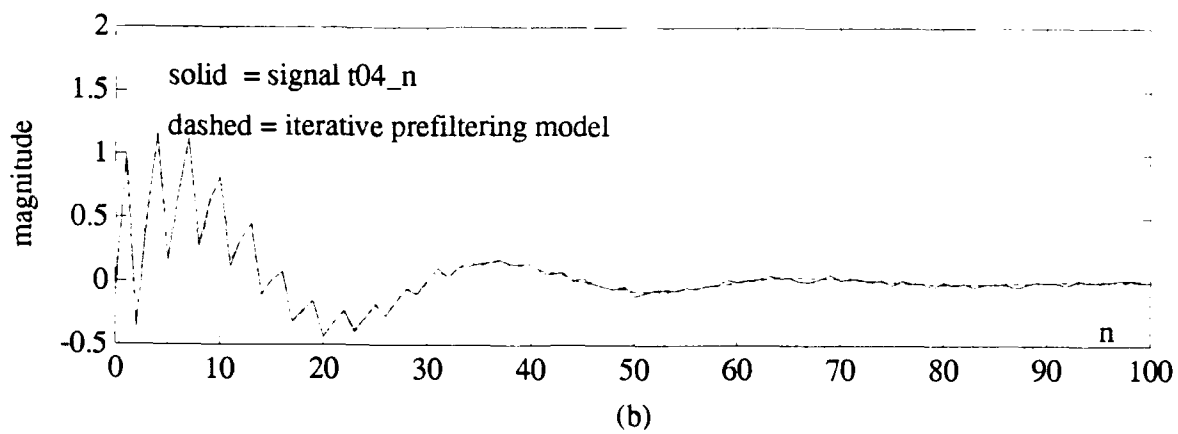
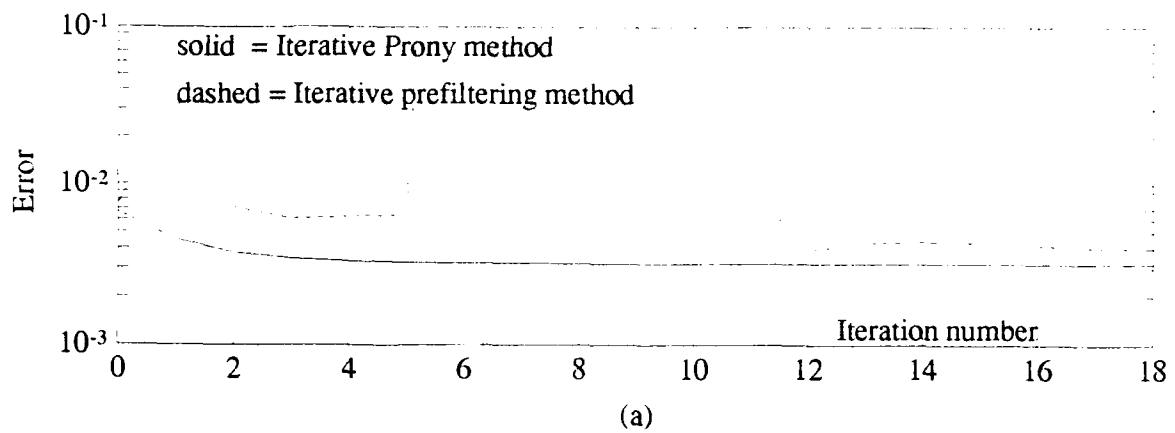


Figure 4.8: Signal $t04_n$ and its **6 poles-5 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t04_n$ and the iterative prefiltering model. (c) Signal $t04_n$ and the iterative Prony model.

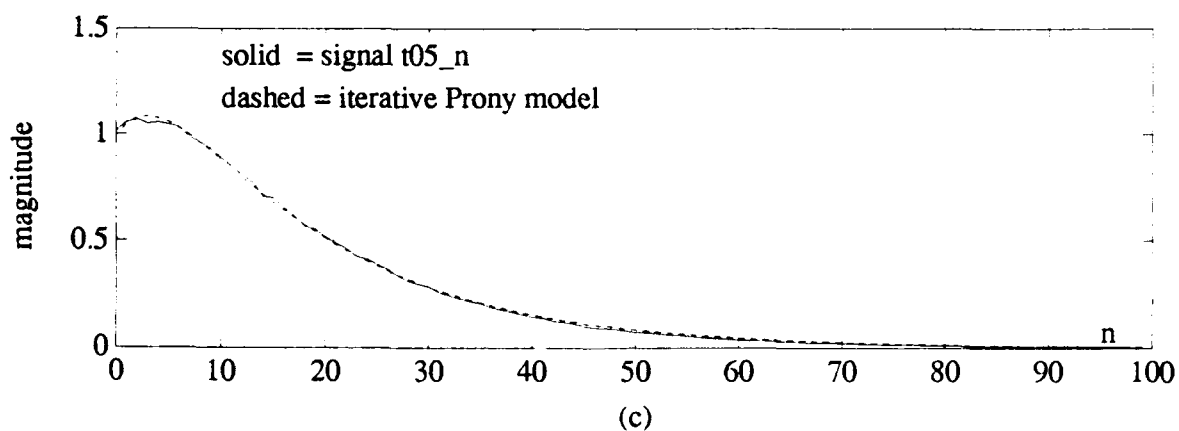
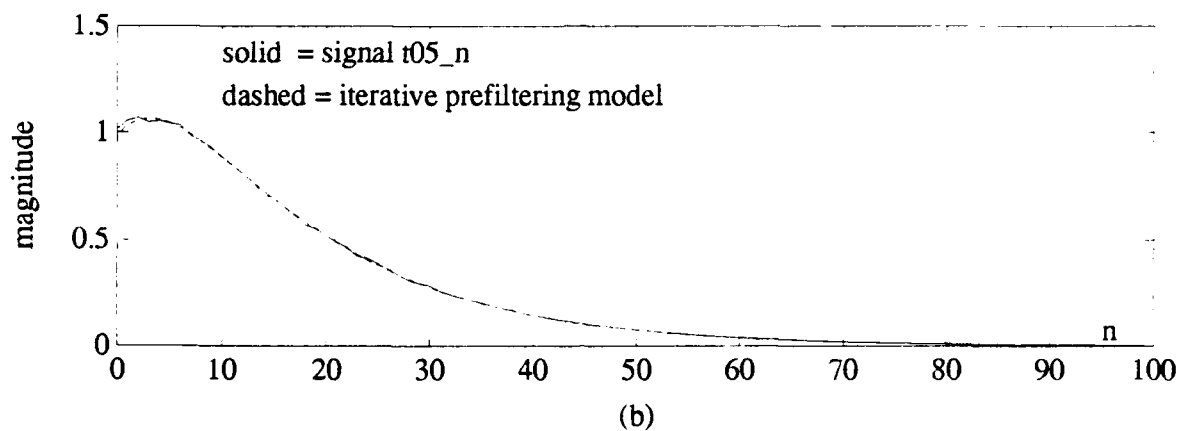
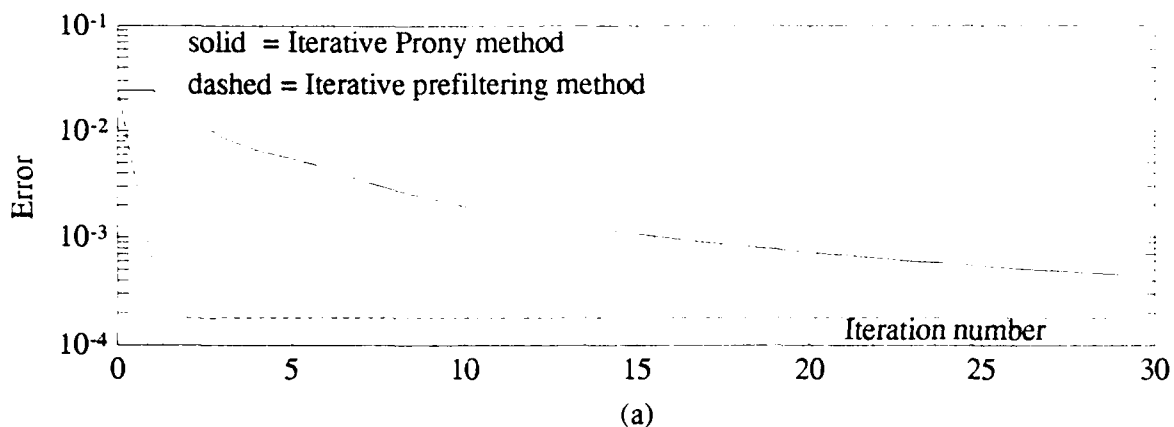


Figure 4.9: Signal $t05_n$ and its **2 poles-1 zero** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t05_n$ and the iterative prefiltering model. (c) Signal $t05_n$ and the iterative Prony model.

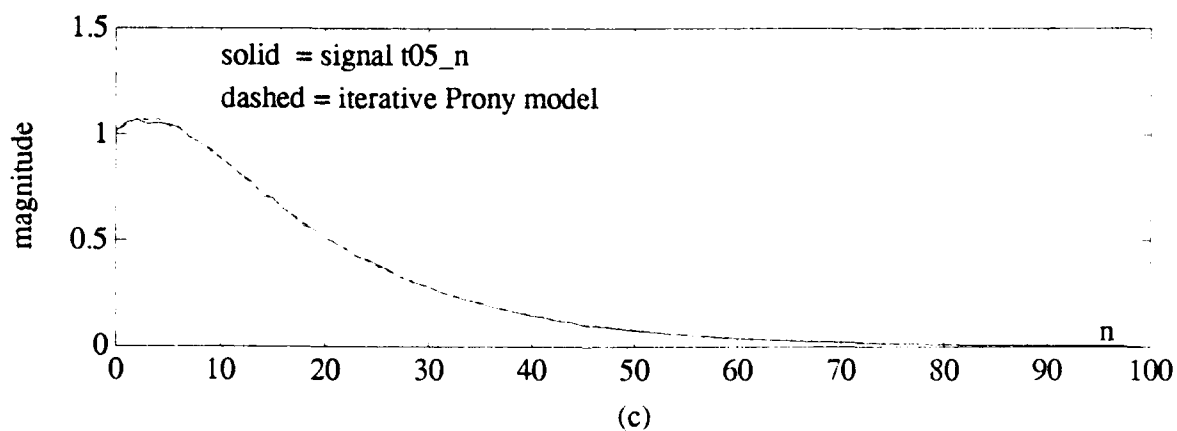
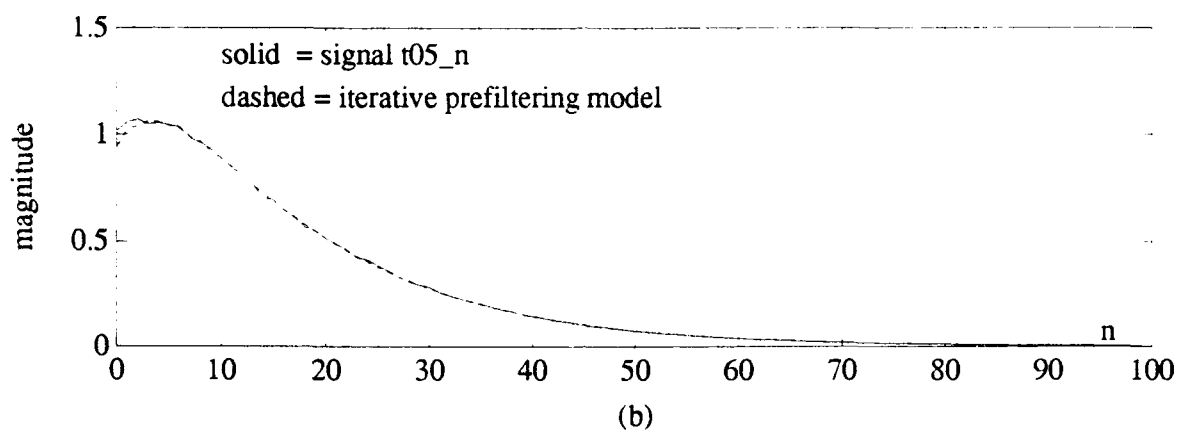
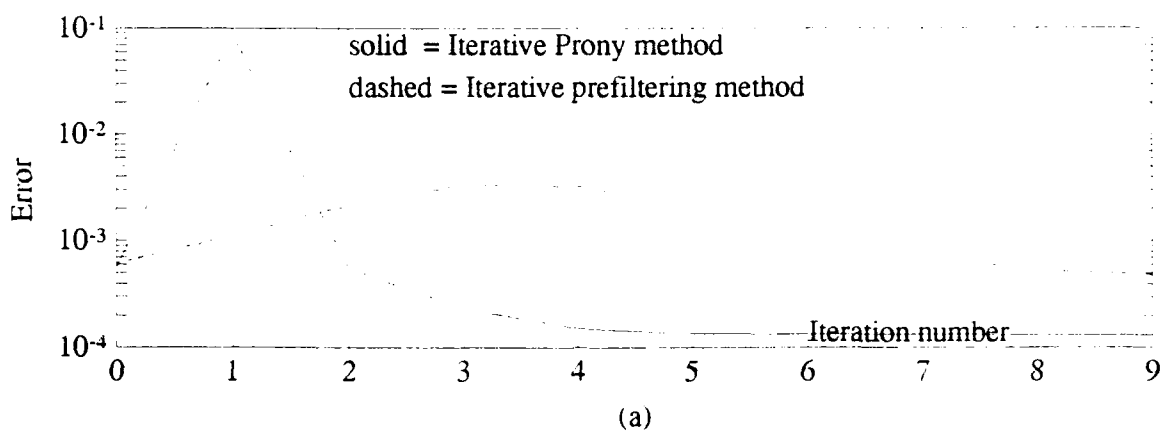


Figure 4.10: Signal $t05_n$ and its **4 poles-3 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal $t05_n$ and the iterative prefiltering model. (c) Signal $t05_n$ and the iterative Prony model.

in this case), the behavior of the iterative prefiltering method tends to degrade (i.e., it takes longer for the algorithm to reach convergence) while the behavior of the iterative Prony method remains the same or even improves as in the case of $t05_n$ shown in Figures 4.9(a) and 4.10(a). Parts (b) and (c) of Figures 4.1 through 4.10 show the original signals and their respective models overlaid. It can be seen that the models arrived at by both methods follow the original signals very closely in all cases. Table 4.3 lists the location of the poles and zeros of the systems used to model all the simulated noisy signals. These systems were obtained using both the iterative prefiltering and the iterative Prony methods. The poles and zeros shown in Table 4.3 can be compared to the poles and zeros of the original simulated signals presented in Table 4.1. It is clear that the location of the poles and zeros of the modeled signals should not be exactly the same as those of the original signals because some noise (in the order of 10 to 15 dB SNR) was intentionally added before the modeling process. However, Tables 4.1 and 4.3 show a close relation between the location of the poles and zeros of the original sequences and the position of the poles and zeros of the modeled signals.

2. Acoustic test data

As mentioned in the last section the acoustic test data represents sounds recorded both underwater and in a laboratory environment. In some cases shorter segments were selected for modeling due to the complexity of these signals. Once again the iterative prefiltering algorithm was used, and its results were compared to the results obtained using the iterative Prony method.

Before presentation of results, it is important to explain how the model produced by the iterative prefiltering method was selected. For all the cases, the same number of iterations was used both for the iterative Prony and for the iterative prefiltering methods. In order to obtain the best possible results from the iterative

TABLE 4.3: POLE-ZERO LOCATION OF THE SYSTEMS USED TO MODEL THE *SIMULATED* NOISY TEST DATA

SIGNAL NAME & (ORDER)	ITERATIVE PRONY		ITERATIVE PREFILTERING	
	POLES	ZEROS	POLES	ZEROS
t01_n (2,1)	0.9511 $\angle \pm 0.6291$	0: 0.7612	0.9507 $\angle \pm 0.6290$	0: 0.7619
t01_n (4,3)	0.9510 $\angle \pm 0.6290$	0.7636	0.9506 $\angle \pm 0.6289$	0.7630
	0.9217 $\angle \pm 3.1397$	0.9476: 0.9151	0.9867 $\angle \pm 2.0944$	0.9886 $\angle \pm 2.0880$
t02_n (4,3)	0.9545 $\angle \pm 0.6238$	13.2102	0.9250 $\angle \pm 0.6313$	6.8483
	0.8455 $\angle \pm 0.6743$	2.1568: 0.1799	0.9142 $\angle \pm 0.6238$	1.6619: 0.4650
t02_n (6,5)	0.9541 $\angle \pm 0.6228$	17.1527	0.9301 $\angle \pm 0.6335$	6.5260
	0.8505 $\angle \pm 0.6797$	2.1237: 0.0952	0.9068 $\angle \pm 0.6173$	1.8203: 0.5542
	0.9896 $\angle \pm 2.7699$	0.8760 $\angle \pm 2.7876$	0.9221 $\angle \pm 2.4366$	0.9800 $\angle \pm 2.3896$
t03_n (4,3)	0.9510 $\angle \pm 1.8850$	22.9950	0.9499 $\angle \pm 1.8851$	12.2468
	0.9505 $\angle \pm 2.3029$	1.0511: 0.9365	0.9514 $\angle \pm 2.3030$	0.9680: 0.8020
t03_n (6,5)	0.9509 $\angle \pm 1.8851$	21.1093	0.9492 $\angle \pm 1.8854$	7.5747
	0.9506 $\angle \pm 2.3029$	0.9340: 0.6882	0.9515 $\angle \pm 2.3028$	0.9566: 0.6099
	0.9356 $\angle \pm 0.2779$	1.1445 $\angle \pm 0.3507$	0.9796 $\angle \pm 1.5986$	0.9870 $\angle \pm 1.6035$
t04_n (4,3)	0.9022 $\angle \pm 2.0981$	22.9853	0.9048 $\angle \pm 2.0956$	2362.7
	0.9515 $\angle \pm 0.2097$	0.9494 $\angle \pm 0.8212$	0.9516 $\angle \pm 0.2090$	0.9000 $\angle \pm 0.8090$
t04_n (6,5)	0.9040 $\angle \pm 2.0966$	17.5349	0.9053 $\angle \pm 2.0933$	57.6799
	0.9517 $\angle \pm 0.2094$	0.9104 $\angle \pm 0.8333$	0.9518 $\angle \pm 0.2093$	0.9486 $\angle \pm 0.8189$
	0.7237: 0.7221	0.7099 $\angle \pm 3.1147$	0.9542: 0.8105	0.9461: 0.7803
t05_n (2,1)	0.9416: 0.7840	0.6693	0.9363: 0.8325	0.7236
t05_n (4,3)	0.9366: 0.8345	0.7330	0.9368: 0.8231	0.7019
	0.9571 $\angle \pm 2.8523$	0.9571 $\angle \pm 2.8436$	0.8839 $\angle \pm 2.7995$	0.8998 $\angle \pm 2.7977$

prefiltering algorithm (especially in those cases when convergence was not obtained), the model selected was that corresponding to the iteration with the lowest error between the model and the original sequence. Figure 4.11 is an example of one of the cases when convergence was not reached using the iterative prefiltering method. It can be seen that if the system selected is the one obtained at the last iteration (20th of the algorithm (which in this case corresponds to a peak of the error)), then the resulting model does not follow the original signal at all. However, if we select the system from the iteration for which the error is the smallest (17th iteration in this case), then we obtain a model that is closer to the original signal. Some insight can also be obtained if we look at the behavior of the poles and zeros of the system while the error of the iterative prefiltering algorithm is oscillating. Figures 4.12 and 4.13 show the position of the poles and zeros during the oscillation that exists between iterations 15 to 20 of the case presented in Figure 4.11. For the first part of these iterations when the error is low, the poles remain at approximately the same position. At iteration 20, (Fig. 4.12(f)) the poles suddenly spread apart, some to even outside the unit circle, causing the large increase in the error. A similar effect occurs with the zeros, although there is some significant movement of the zeros in the earlier iterations. This type of behavior is avoided in the iterative Prony method where a controlled and systematic displacement of the poles and zeros is produced to finally reduce the error between the modeled and original signals.

The same approach used in the last subsection was used to model the *acoustic* data. All signals were first modeled with a low order system and subsequently remodeled using a higher order system. The results are shown in Figures 4.14 to 4.23. In most of the cases the models obtained using the iterative Prony method closely follow the original signals. It can also be observed that the iterative Prony method does not have as large a dependency in the order of the model selected as does the iterative prefiltering method. While it is of course necessary to select a model with a

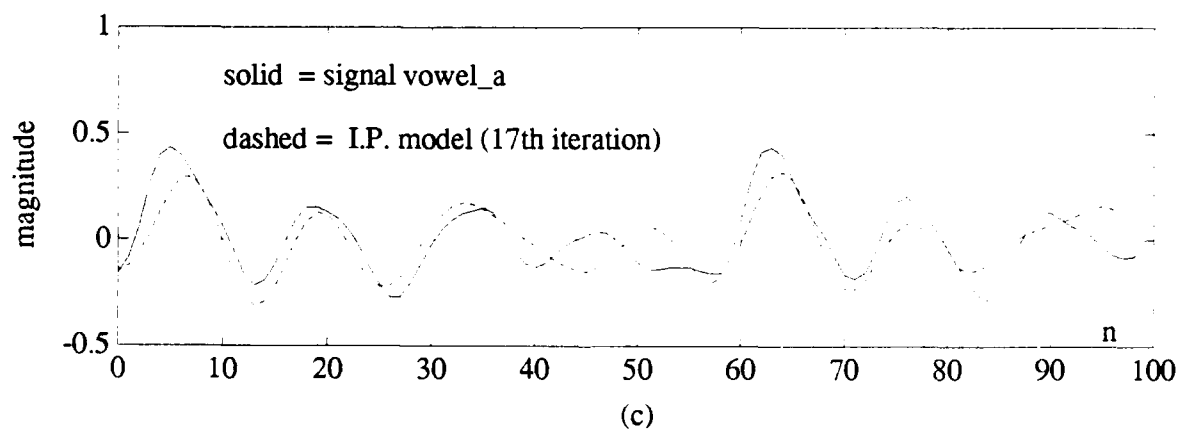
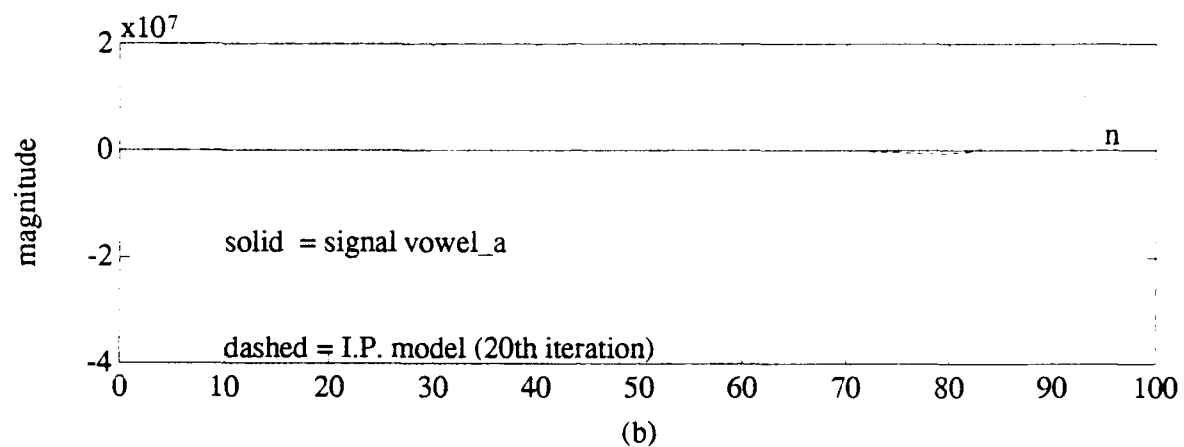
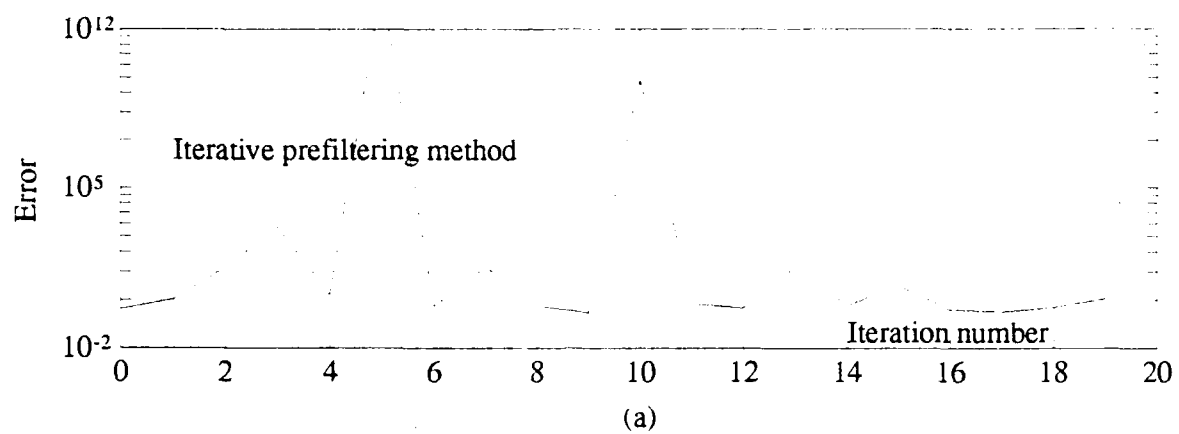


Figure 4.11: Signal *vowel_a* being modeled by iterative prefiltering using a (10.9) order system. (a) Normalized squared-norm of the error between the model and the actual signal. (b) Signal *vowel_a* and the iterative prefiltering model selected from the 20th iteration. (c) Signal *vowel_a* and the iterative prefiltering model selected from the 17th iteration.

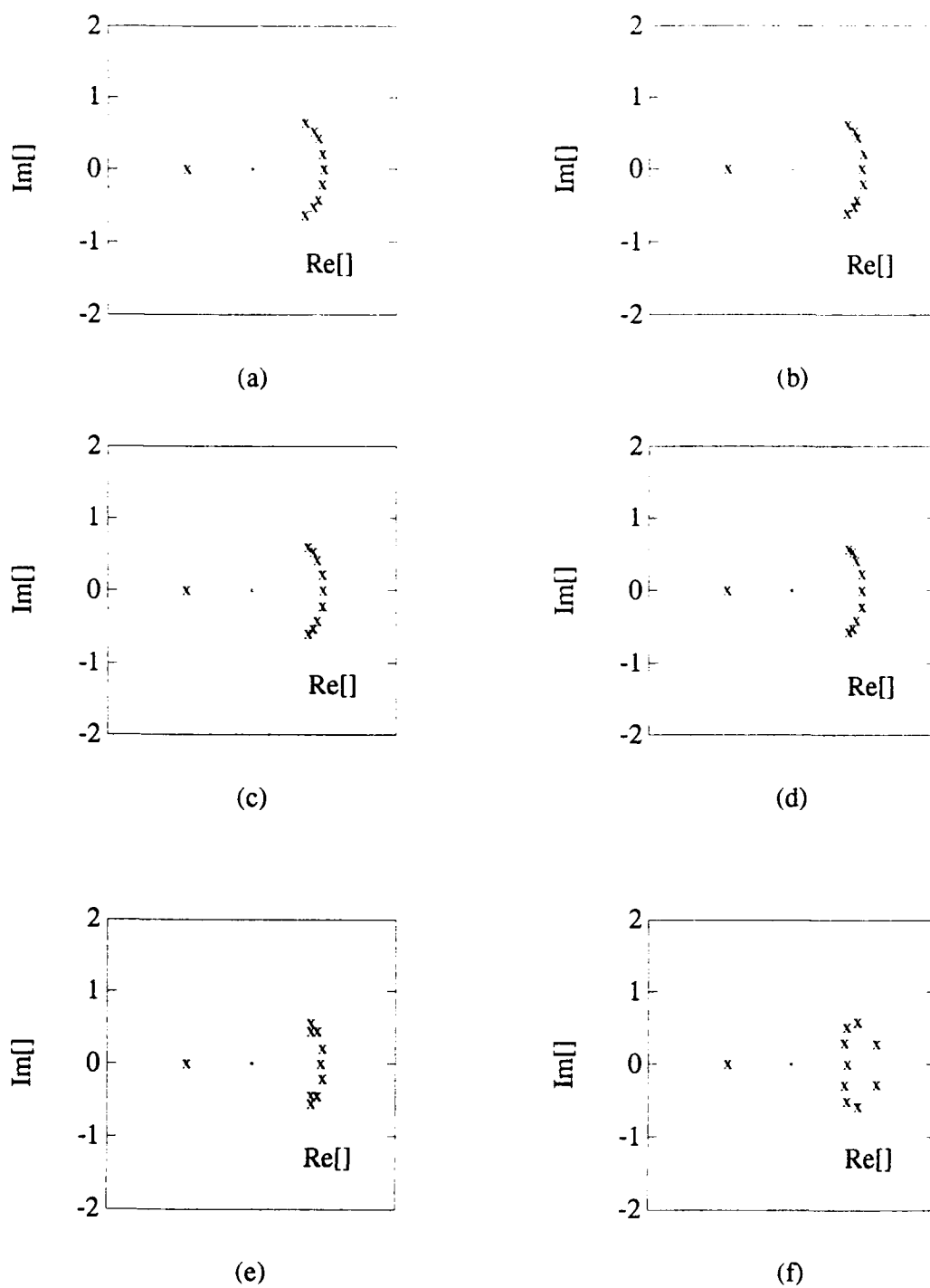


Figure 4.12: Behavior of the poles of the system used to model the signal *vowel_a* during one oscillation of the iterative prefiltering algorithm. (a) 15th iteration. (b) 16th iteration. (c) 17th iteration. (d) 18th iteration. (e) 19th iteration. (f) 20th iteration.

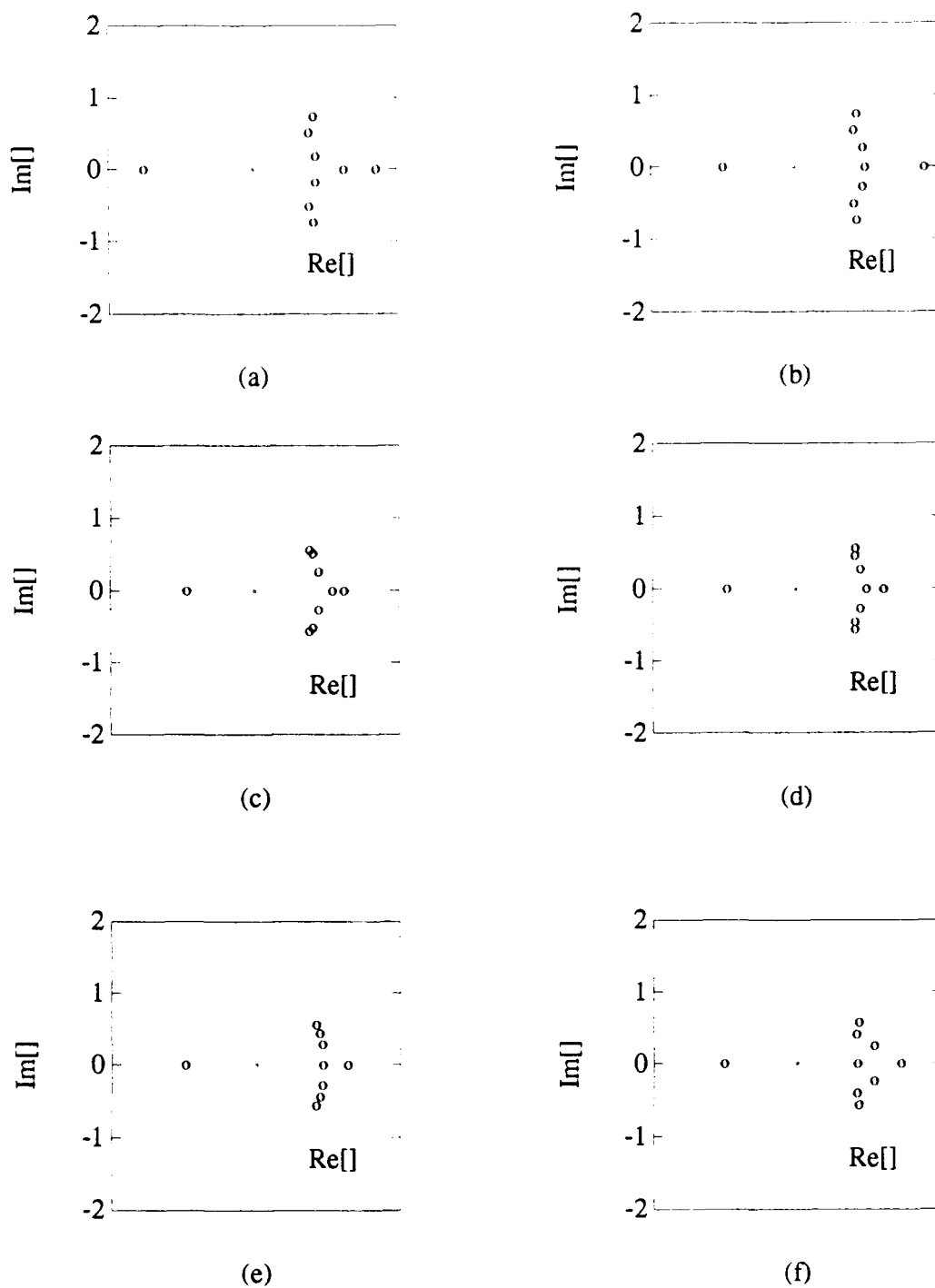


Figure 4.13: Behavior of the zeros of the system used to model the signal *rowel_a* during one oscillation of the iterative prefiltering algorithm. (a) 15th iteration. (b) 16th iteration. (c) 17th iteration. (d) 18th iteration. (e) 19th iteration. (f) 20th iteration.

high enough order to properly model the original signal, once such a model has been selected, increments or small variations to its order do not produce degradations on the performance of the algorithm. On the contrary, it was found that in some cases the performance of the iterative prefiltering algorithm can be significantly reduced when the order of the model is increased slightly. It can also be seen that the iterative Prony algorithm tends to provide a closer match to the data than the iterative prefiltering method, and in most of the cases the rate of convergence of the iterative Prony method was higher than that of iterative prefiltering. Another important point that can be extracted from the results presented in Figures 4.14 to 4.23 is that while convergence with neither algorithm is guaranteed, we obtained convergence with the iterative Prony method in all cases for these acoustic signals. The same was not true for iterative prefiltering. In most of the cases the error for the iterative Prony method begins to decrease starting at the first iteration, and although the change is not monotonic in all cases, the error after a few iterations is consistently lower than the initial error.

(This page intentionally left blank)

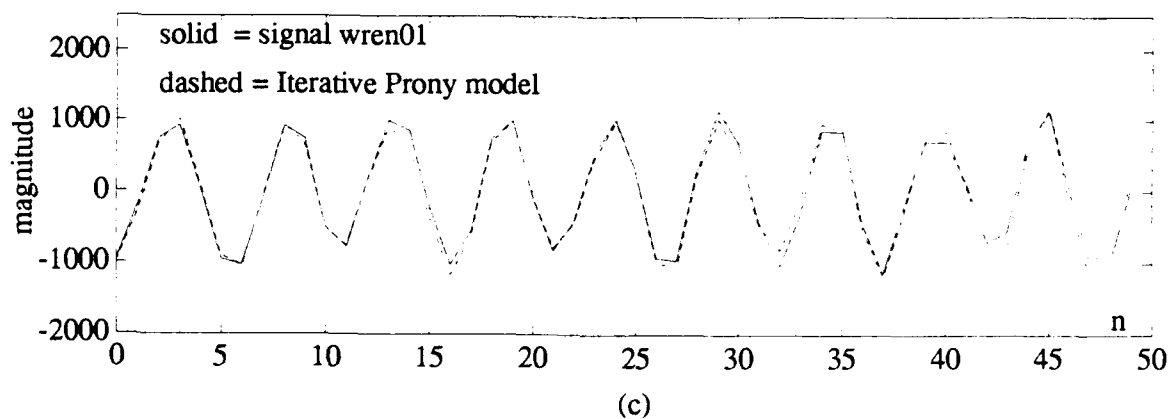
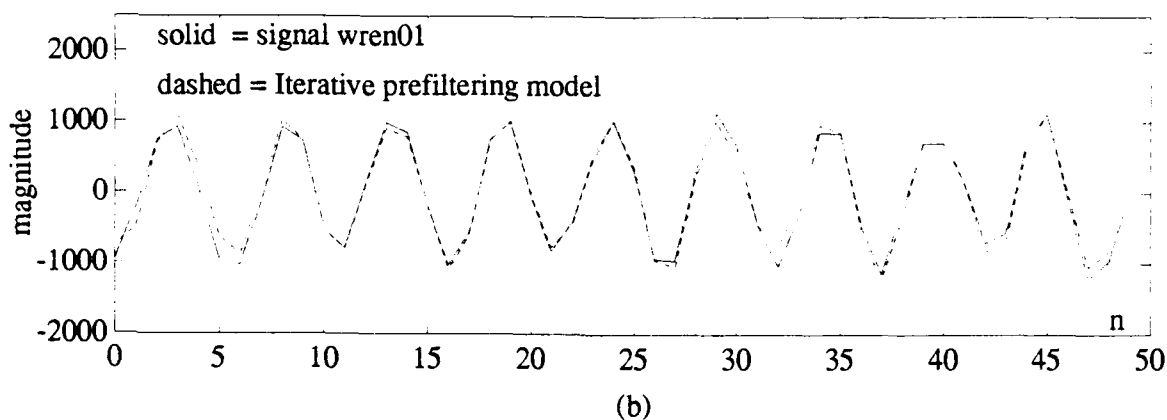
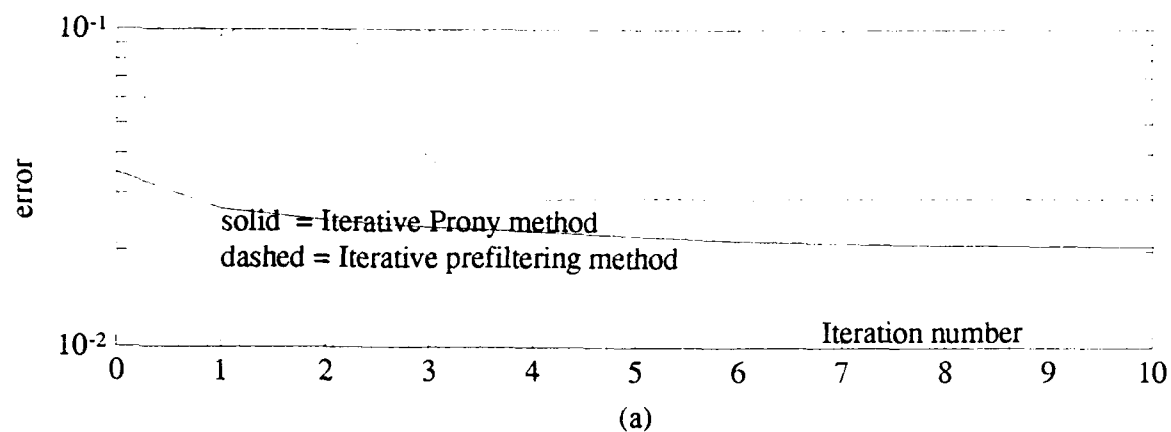
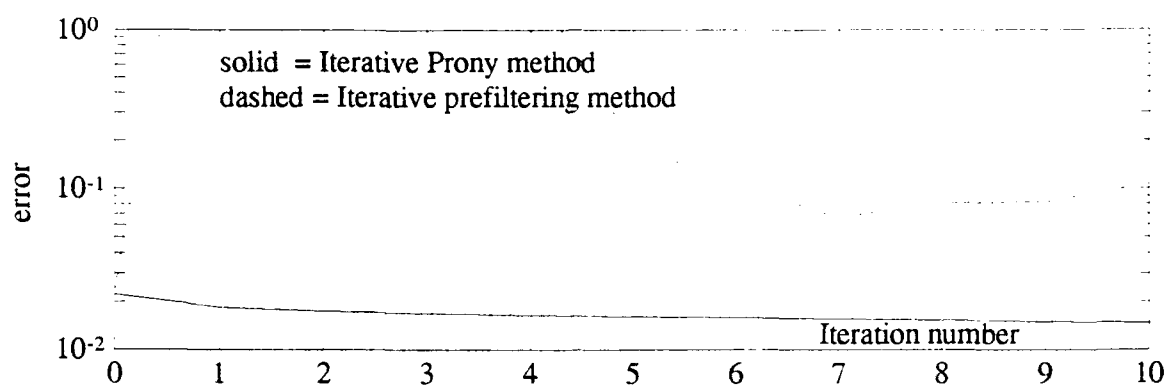
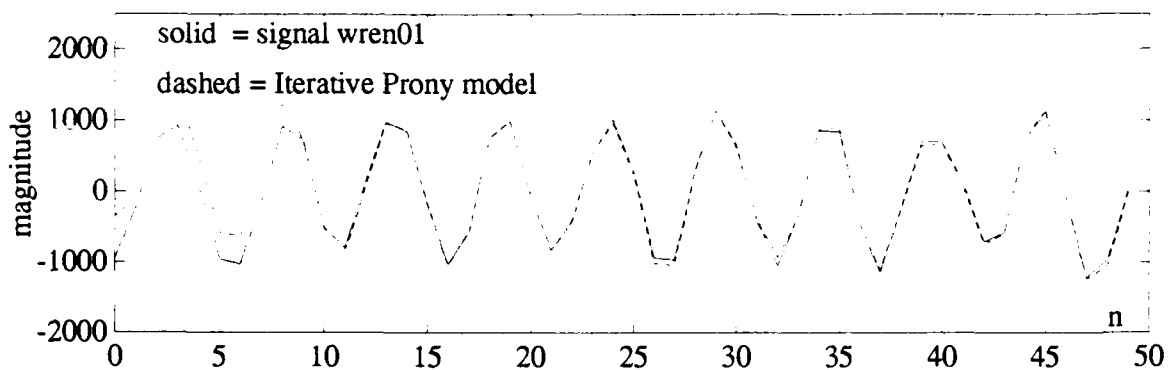


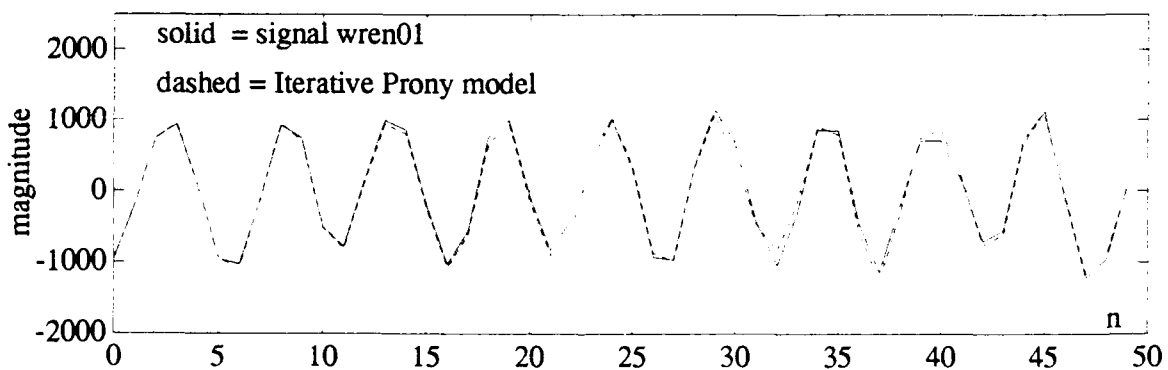
Figure 4.14: Signal *wren01* and its **7 poles-6 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *wren01* and the iterative prefiltering model. (c) Signal *wren01* and the iterative Prony model.



(a)



(b)



(c)

Figure 4.15: Signal *wren01* and its **12 poles-11 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *wren01* and the iterative prefiltering model. (c) Signal *wren01* and the iterative Prony model.

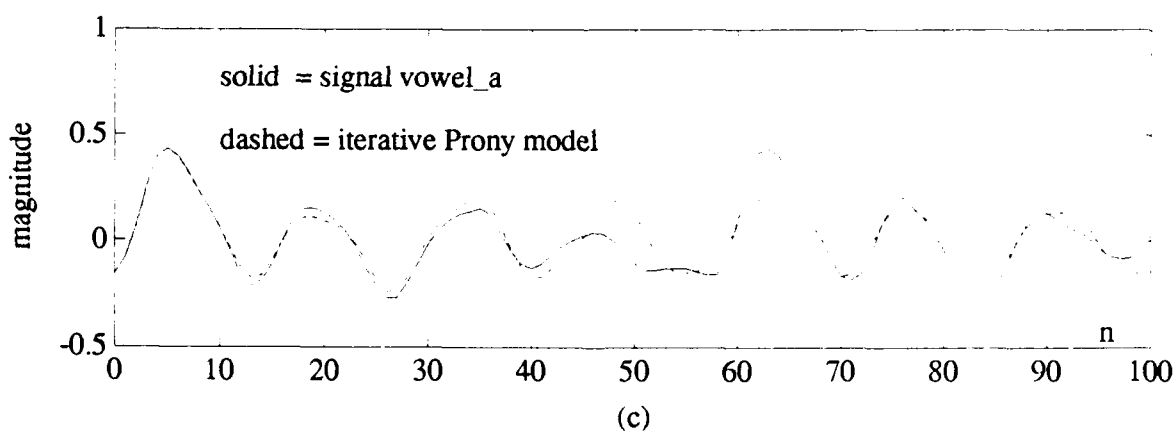
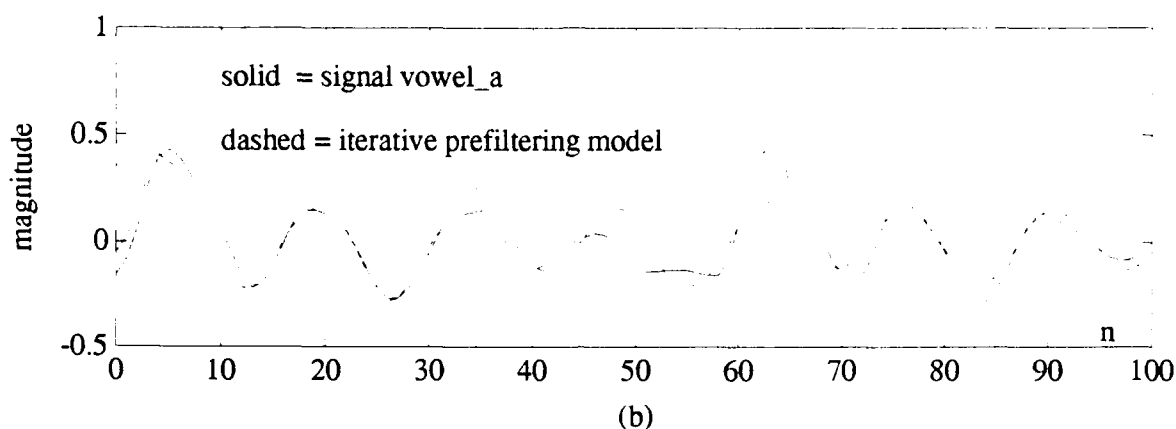
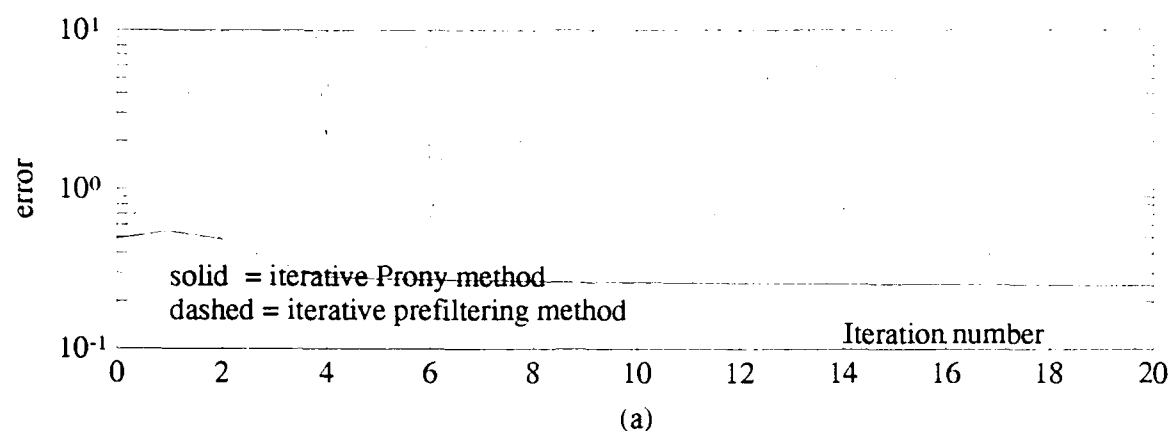


Figure 4.16: Signal *vowel_a* and its **10 poles-9 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *vowel_a* and the iterative prefiltering model. (c) Signal *vowel_a* and the iterative Prony model.

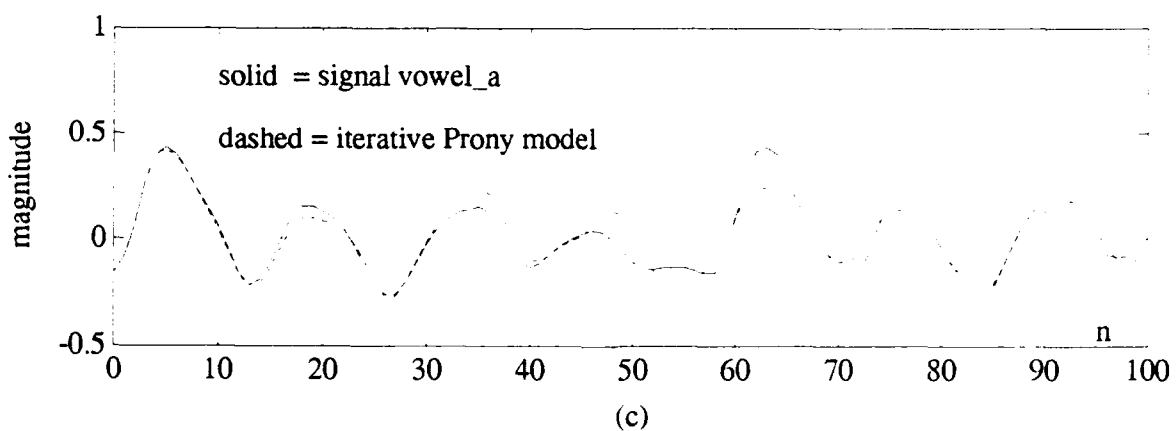
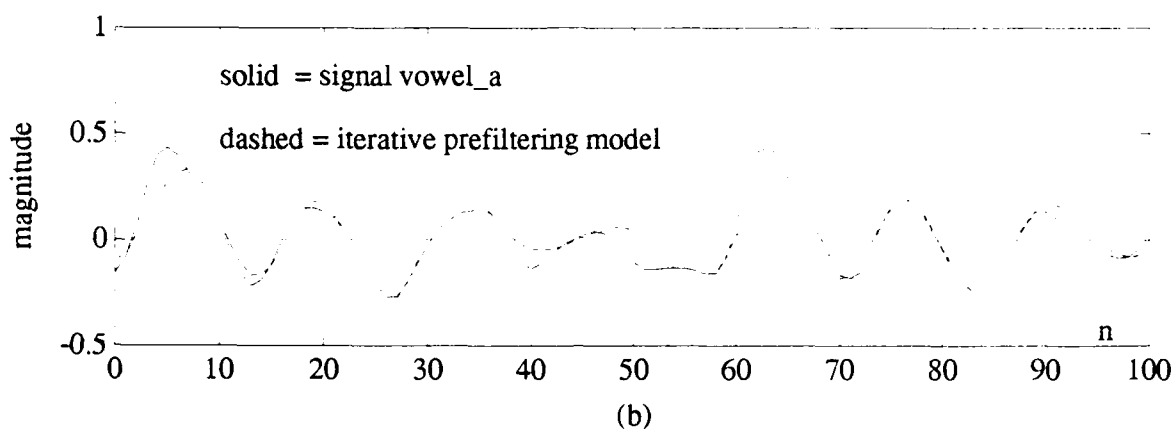
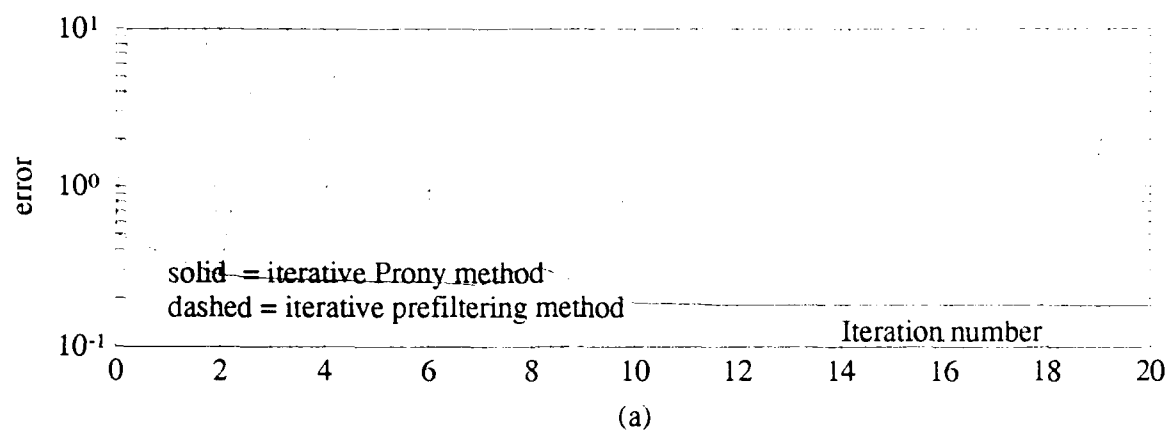


Figure 4.17: Signal *vowel_a* and its **14 poles-13 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *vowel_a* and the iterative prefiltering model. (c) Signal *vowel_a* and the iterative Prony model.

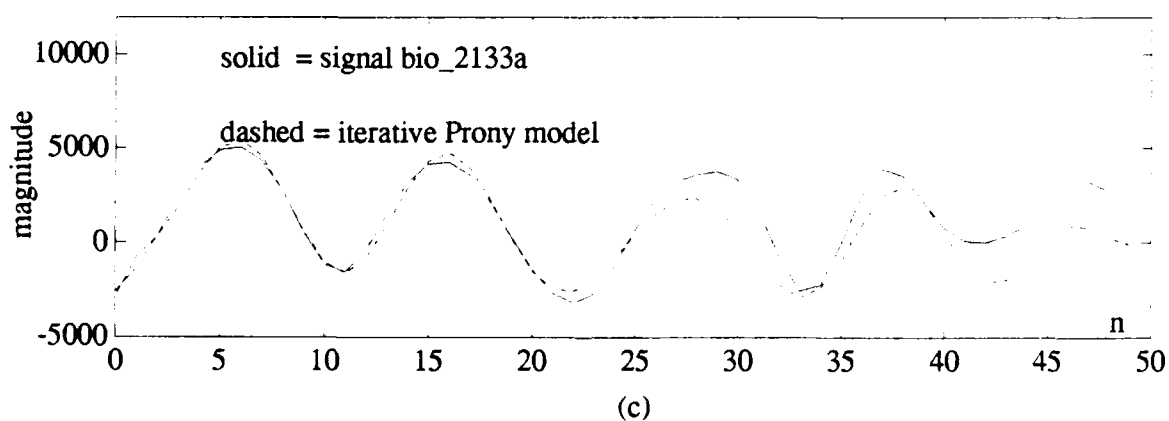
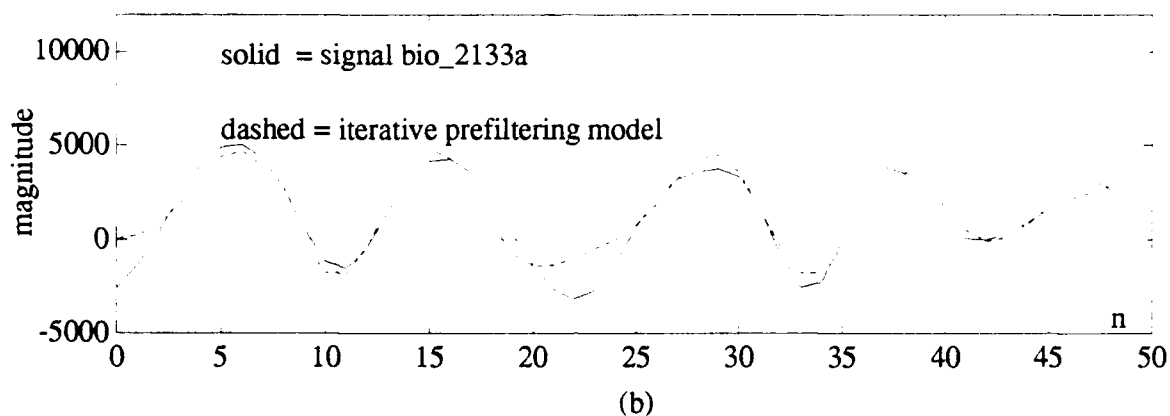
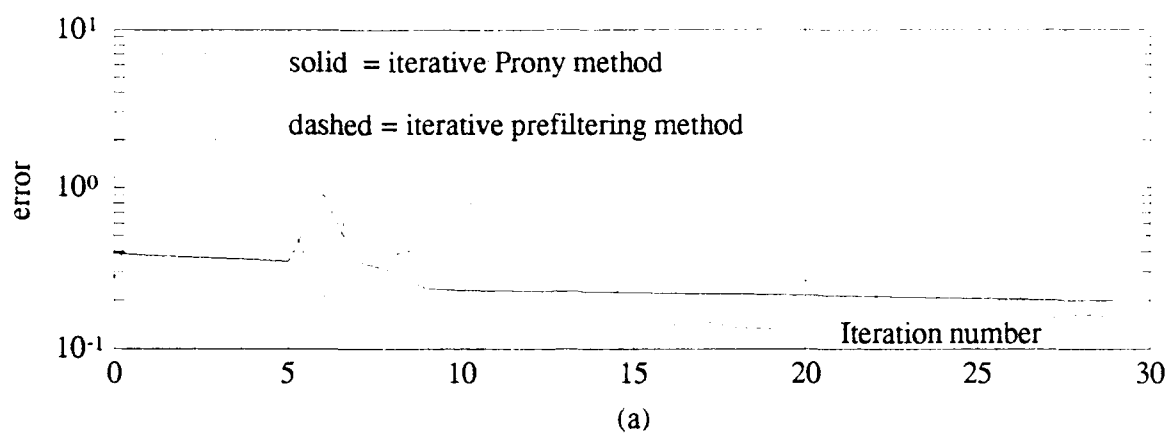


Figure 4.18: Signal *bio_2133a* and its **8 poles-7 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *bio_2133a* and the iterative prefiltering model. (c) Signal *bio_2133a* and the iterative Prony model.

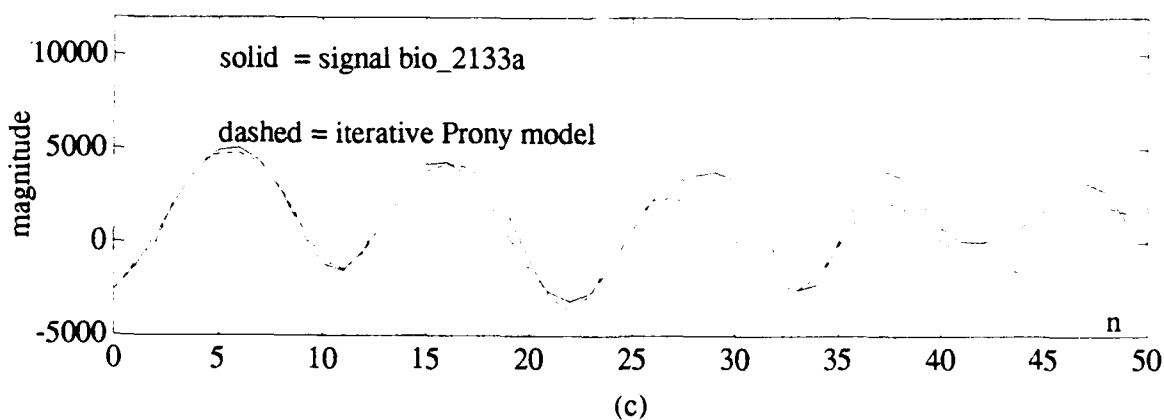
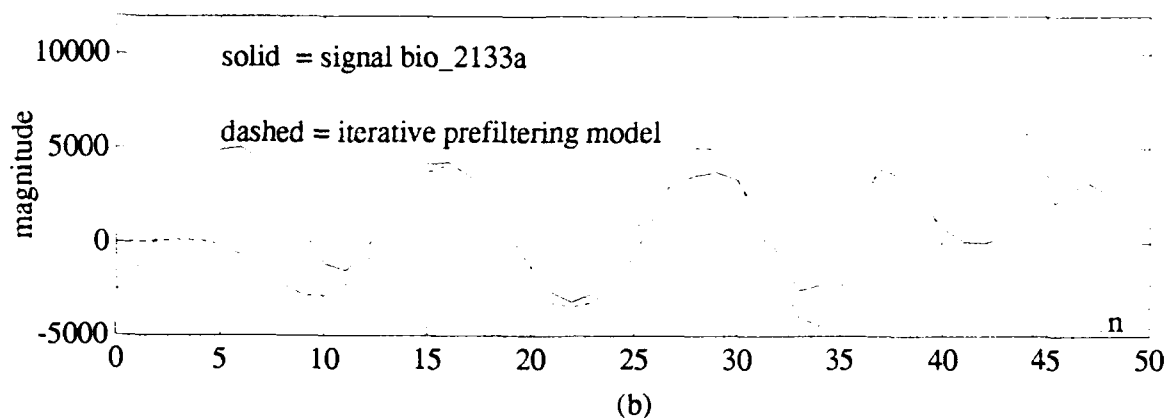
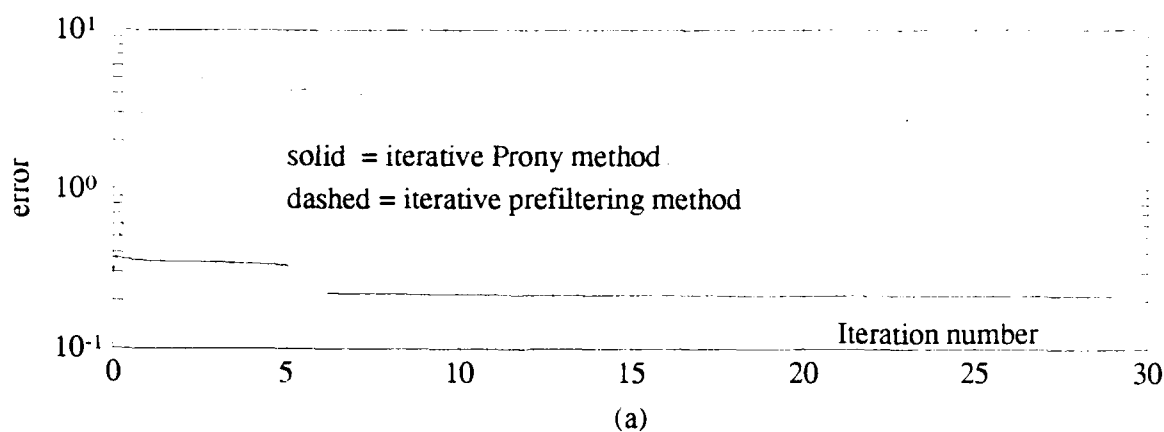


Figure 4.19: Signal *bio_2133a* and its **12 poles-11 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *bio_2133a* and the iterative prefiltering model. (c) Signal *bio_2133a* and the iterative Prony model.

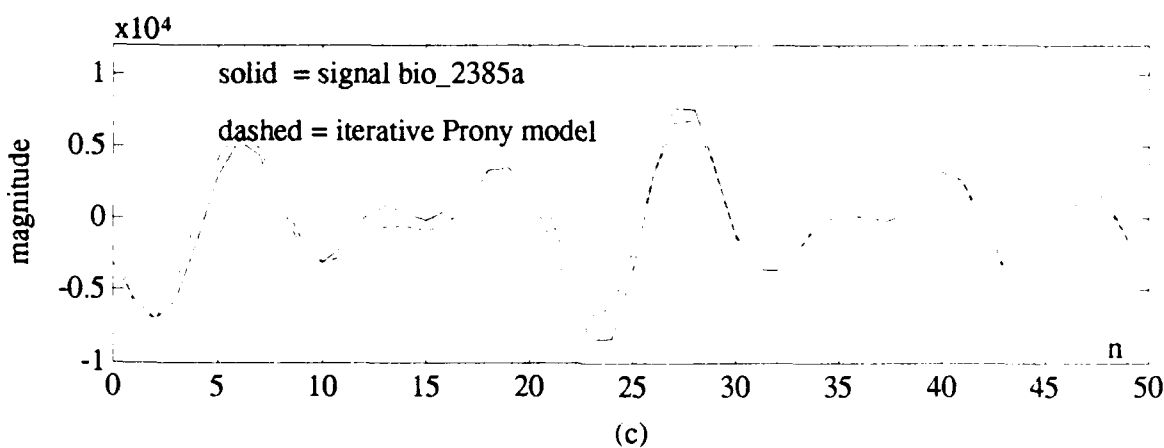
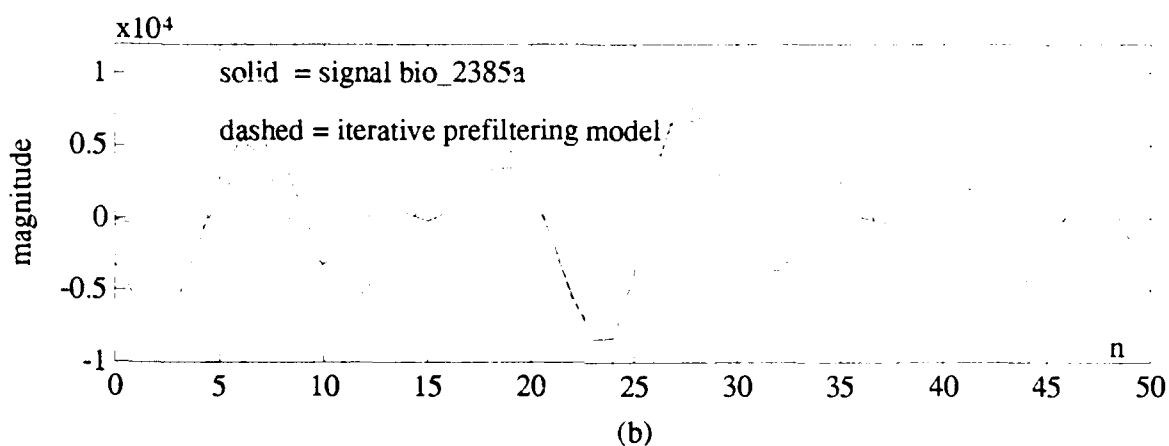
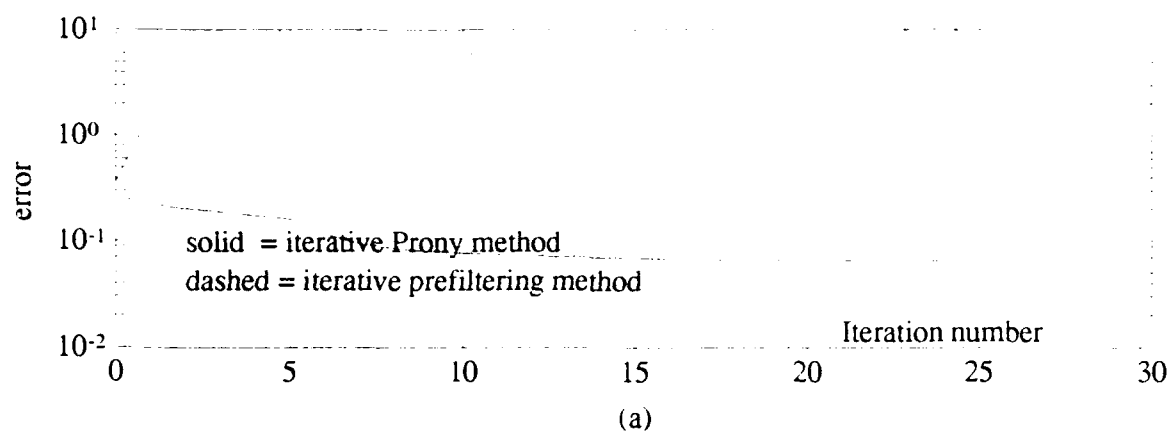


Figure 4.20: Signal *bio_2385a* and its **8 poles-7 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *bio_2385a* and the iterative prefiltering model. (c) Signal *bio_2385a* and the iterative Prony model.

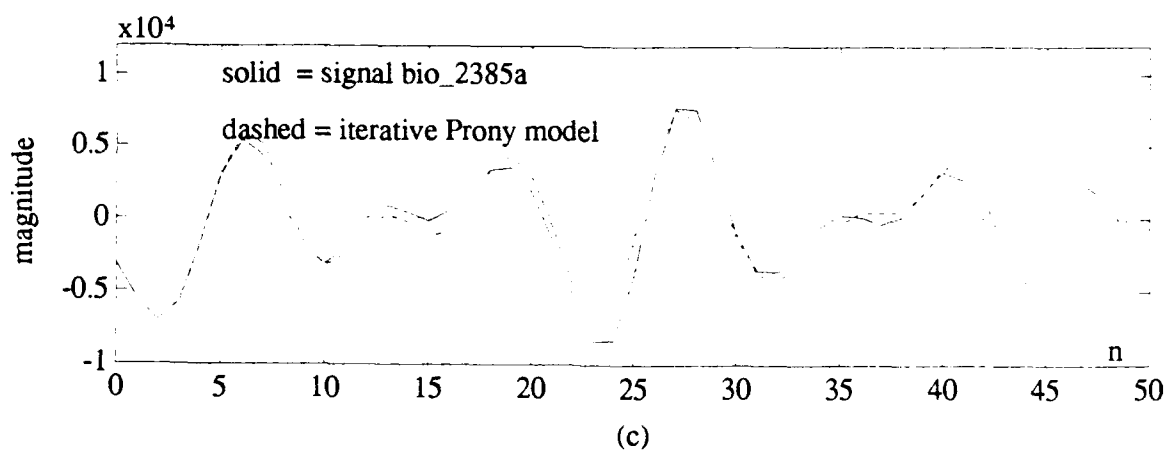
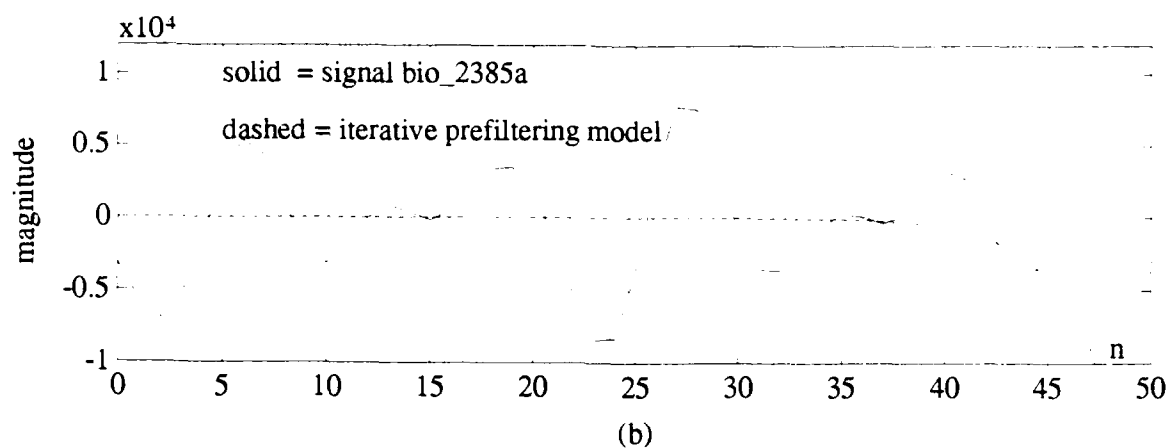
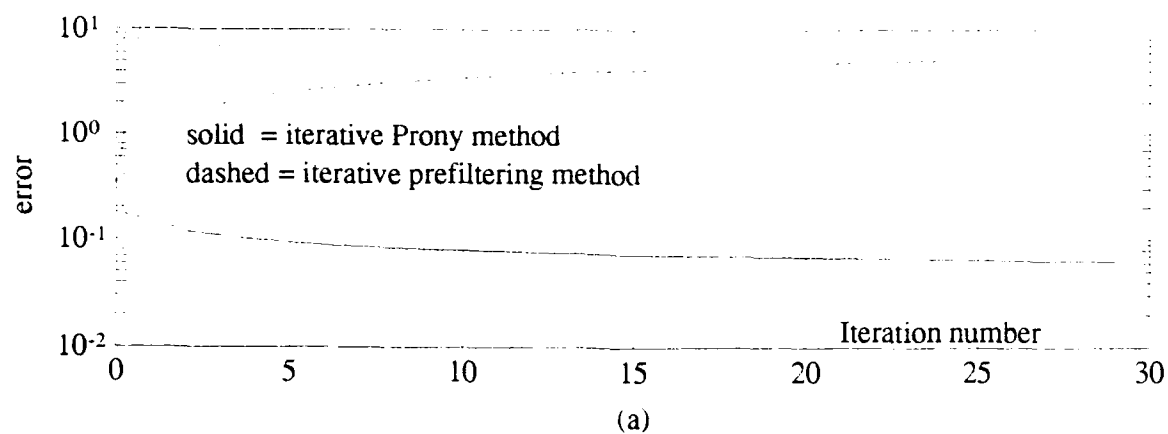


Figure 4.21: Signal *bio_2385a* and its **12 poles-11 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *bio_2385a* and the iterative prefiltering model. (c) Signal *bio_2385a* and the iterative Prony model.

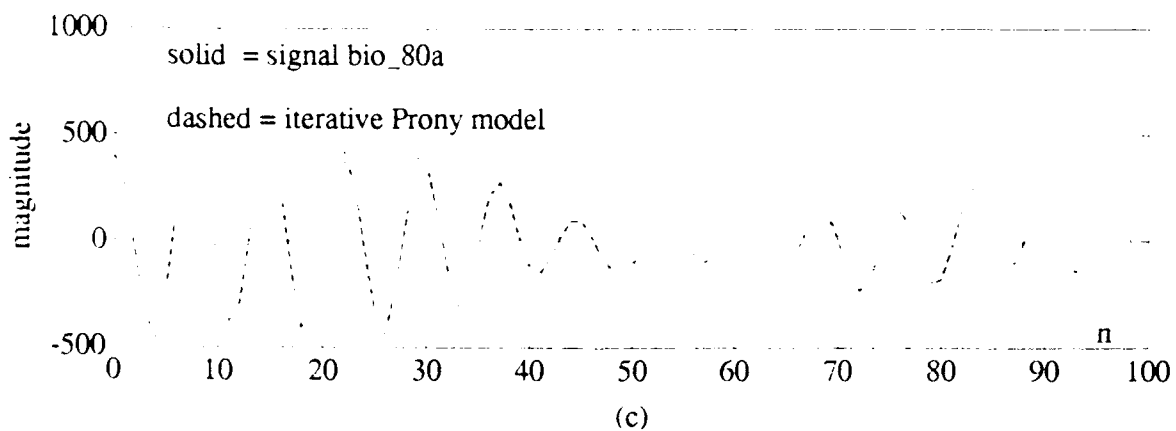
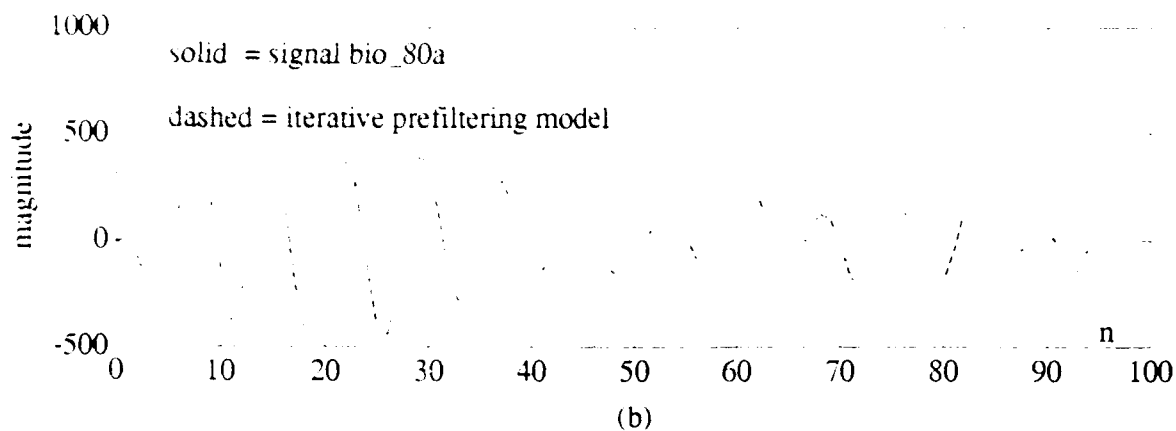
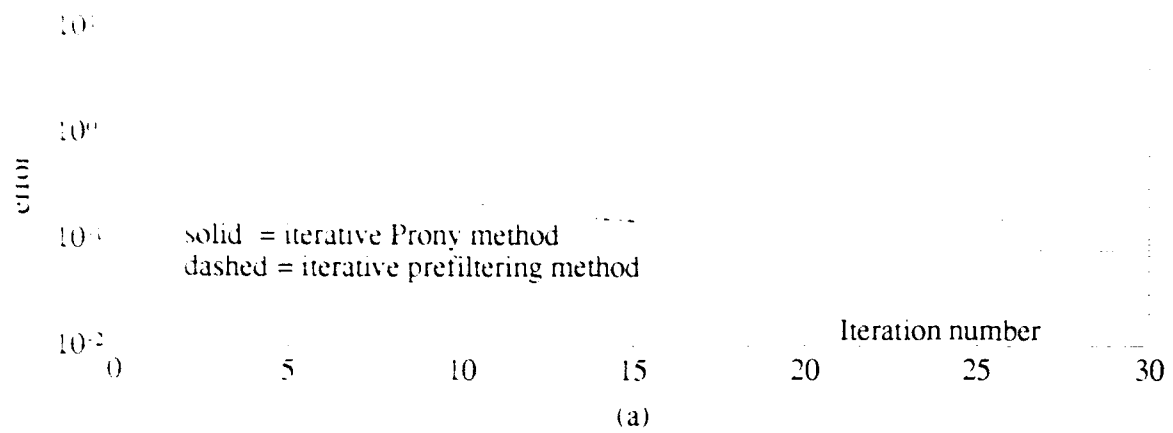


Figure 4.22: Signal *bio_80a* and its **8 poles-7 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *bio_80a* and the iterative prefiltering model. (c) Signal *bio_80a* and the iterative Prony model.

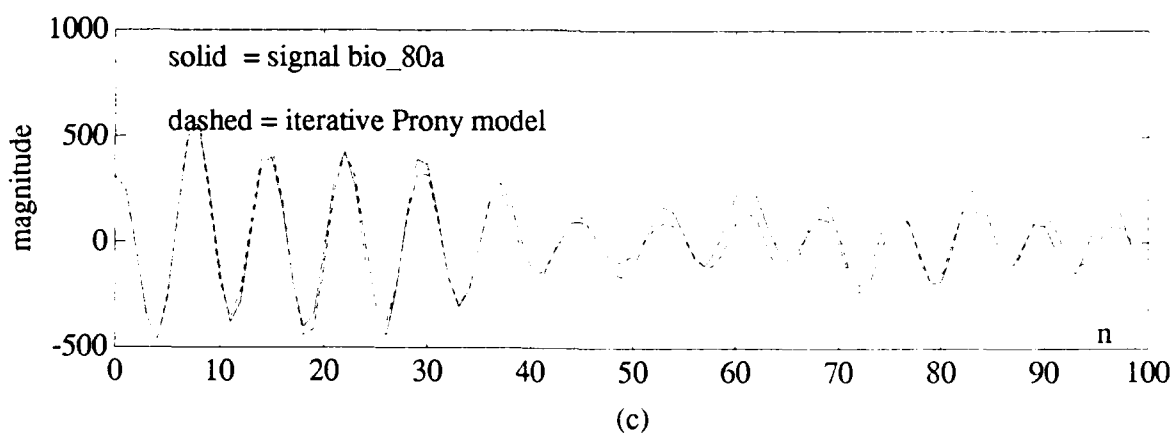
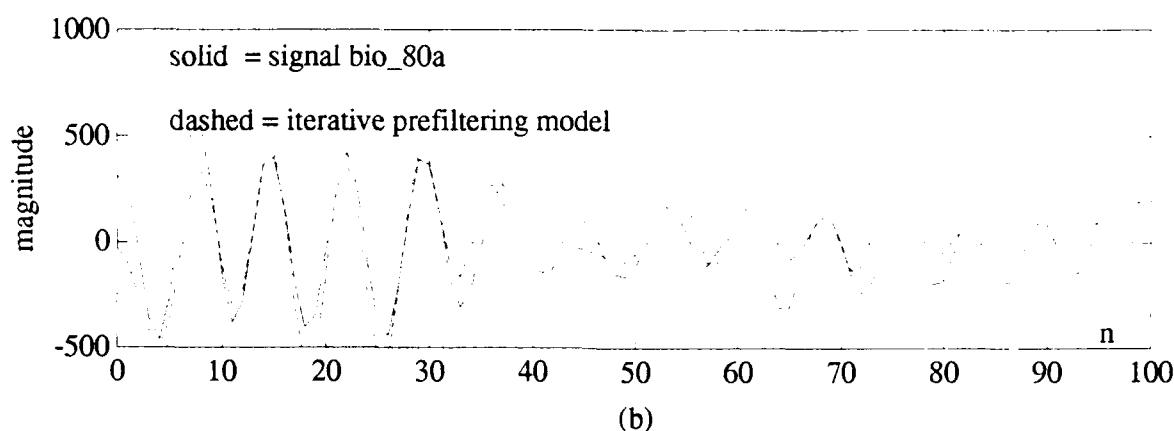
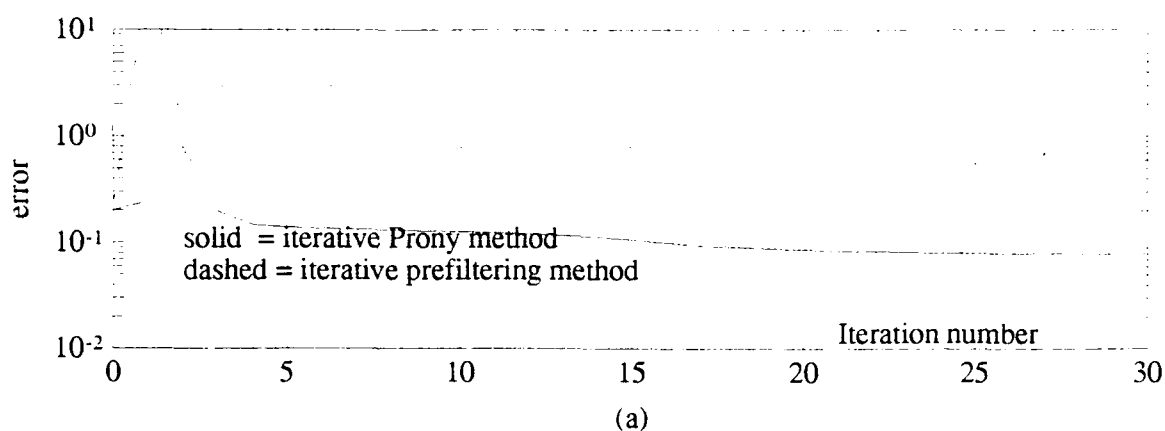


Figure 4.23: Signal *bio_80a* and its **12 poles-11 zeros** models. (a) Normalized squared-norm of the error between the models and the actual signal. (b) Signal *bio_80a* and the iterative prefiltering model. (c) Signal *bio_80a* and the iterative Prony model.

V. CONCLUSIONS

A. DISCUSSION OF RESULTS

In this thesis, a new method for modeling signals in the time domain is developed and applied to model both recorded acoustic data and simulated signals produced as the impulse response of a known system. We call this method the *iterative Prony* method. In most of the simulated test data sets the models provided by the iterative Prony method are sufficiently close to the original signals; in most cases, it is difficult to distinguish between the signal and the model. When modeling the acoustic data distortion becomes apparent in some of the models, which may be due to the complexity of the structure of the signals. However, this distortion is no worse than for any of the best algorithms that have been used to model this data previously.

The new algorithm was compared very specifically to the iterative prefiltering algorithm [Ref. 7, 8] which has been used in modeling a variety of acoustic data [Ref. 3, 9]. The rate of convergence of the iterative Prony method was in most of the cases comparable or superior to that of the iterative prefiltering algorithm. Thus, while iterative prefiltering sometimes has convergence problems, the new algorithm is much more dependable in that respect. The price to pay for this improvement is in the number of computations. While the number of floating point operations per iteration in the iterative prefiltering method is approximately

$$64(P + Q - 1)^3 + 8N_s(P + Q)^2 + 10(P + Q)N_s + 12N_s,$$

iterative Prony requires about

$$672P^3 + (24N_s + 102)P^2 + (60N_s + 46)P + 198N_s$$

floating point operations at each iteration. For example for a complex data set of length 100 ($N_s = 100$) and $P = Q = 8$ we have approximately 452,400 floating point operations per iteration using iterative prefiltering versus 572,360 using iterative Prony. If we increase the order of the model to $P = Q = 16$, then we have approximately 2,787,824 operations per iteration in the iterative prefiltering algorithm versus 3,509,560 in the iterative Prony method.

B. RECOMMENDATIONS FOR FUTURE WORK

The iterative prefiltering algorithm has been the main tool in the modeling efforts for sonar data modeling [Ref. 9, 13]. The new iterative Prony algorithm is now at a stage where it can be substituted for the iterative prefiltering algorithm and tested in operational use. To do so needs some further programming to make the segmentation of the data automatic and to make the entire modeling procedure more of a "turn crank" operation. These should be some of the very next steps. In addition, the practical implications of the increased computation needs to be addressed, and if possible new methods need to be developed to help reduce computations.

In a larger sense the work reported in this thesis can be used as a base for possible applications of the iterative Prony method in the problems of filter design, speech processing, and spectral estimation. The expressions for the vector of first derivatives \mathbf{g} and the matrix of second derivatives \mathbf{G} of the error derived in Chapter III can be used along with different minimization methods to provide for other new modeling methods that may adapt better to specific modeling problems.

LIST OF REFERENCES

- [1] Therrien, Charles W., *Discrete Random Signals and Statistical Signal Processing*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1992.
- [2] McClellan, James H., "Parametric Signal Modeling," in *Advanced Topics in Signal Processing*, Lim, Jae S. and Oppenheim, Alan V., pp. 1-57, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1987.
- [3] May, Gary L., *Pole-Zero Modeling of Transient Waveforms: A Comparison of Methods with Application to Acoustic Signals*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1991.
- [4] Box, George E.P. and Jenkins, Gwilym M., *Time Series Analysis: Forecasting and Control*, rev. ed., Holden-Day, Oakland, California, 1976.
- [5] Marple, S. Lawrence Jr., *Digital Spectral Analysis with applications*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1987.
- [6] Kay, Steven M., *Modern Spectral Estimation: Theory and Application*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1988.
- [7] Steiglitz, K. and McBride, L.E., "A technique for the identification of linear systems," *IEEE Transactions on Automatic Control*, AC-10, pp. 461-464, October 1965.
- [8] Steiglitz, K., "On the simultaneous estimation of poles and zeros in speech analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-25, pp. 229-234, June 1977.
- [9] Johnson, Thomas P., *ARMA Modeling Methods for Acoustic Signals*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1992.
- [10] Gottfried, Byron S. and Weisman, Joel., *Introduction to Optimization Theory*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1973.
- [11] Fletcher, Roger, *Practical Methods of Optimization*, Second Edition, John Wiley & Sons, New York, 1987.
- [12] Marquardt, D. W., "An algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal*, Volume 11, No. 2, pp. 431-441, June 1963.

- [13] Victory, Charles W., *Comparison of Signal Processing Modeling Techniques of Passive Sonar Data*. Master's Thesis, Naval Postgraduate School, Monterey, California. (to be published), March 1993.

INITIAL DISTRIBUTION LIST

- | | | |
|----|---|---|
| 1. | Defense Technical Information Center
Cameron Station
Alexandria, VA. 22314-6145 | 2 |
| 2. | Dudley Knox Library, Code 52
Naval Postgraduate School
Monterey, CA 93943-5002 | 2 |
| 3. | Chairman, Code EC
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943 | 1 |
| 4. | Prof. Charles W. Therrien, Code EC/Ti
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943 | 2 |
| 5. | Prof. Murali Tummala, Code EC/Tu
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943 | 1 |
| 6. | Mr. Steve Grenandee, Code 2121
Naval Underwater Systems Center
New London, CT 06320-5594 | 1 |
| 7. | Commander, Naval Sea Systems Command
Attn: Cdr. Thomas Mason (Code 06UR)
Naval Sea Systems Command Headquarters
Washington, D.C. 20362-5101 | 1 |
| 8. | Commander, Naval Air Systems Command
Attn: Mr. Earl Benson (PMA 264, Room 740, JP-1)
Naval Air Systems Command Headquarters
Washington, D.C., 20361-5460 | 1 |

- | | | |
|-----|---|---|
| 9. | Defense Advanced Research Projects Agency
Attn: Mr. Paul Rosenstrach
Suite 600
1555 Wilson Blvd.
Arlington, VA, 22209 | 1 |
| 10. | Chief of Naval Research
Office of the Chief of Naval Research
Attn: Dr. Rabi Madan (Code 1114)
Arlington, VA, 22217-5000 | 1 |
| 11. | Dr. Vivec Samant
ORINCON Corp.
9363 Towne Centre Drive
San Diego, CA, 92121 | 1 |
| 12. | Dr. Lou Griffith
Code 7304
NCCOSC, RDT & E Division
San Diego, CA, 92152-5000 | 1 |